ADMIN Network & Security

VoIP Security | Microsoft Loop | Kata Containers

ADMIN

Network & Security

ISSUE 90

ubuntu
25.10 Server
ISSUE 90/2025
ADMIN
Network & Security
DVD INSIDE

VoIP Network Security

Discover vulnerabilities and boost infrastructure security

MLOps Pipeline

Tetragon
A security suite for Kubernetes

Hotpatching Win Server 2025

Kata Containers
VM isolation for containerized workloads

Microsoft Loop

Administrator Protection
State-of-the-art protection replaces UAC

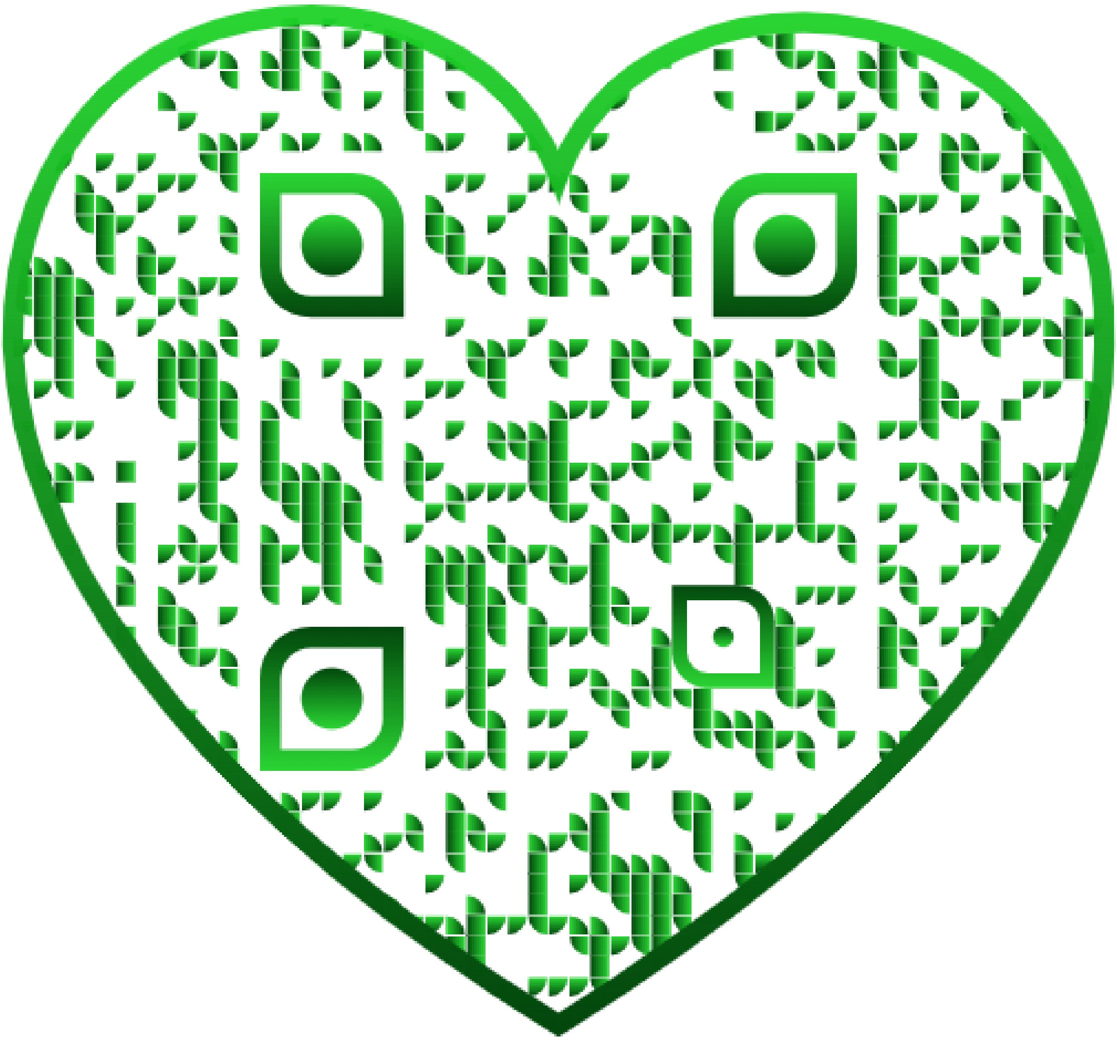
Sigstore
Protection against software supply chain attacks

Puppet Autosign Policies
Automate Debian workstation enrollment

Ollama+Cursor
On-premises open source AI developer workflow

LINUX NEW MEDIA
The Pulse of Open Source

WWW.ADMIN-MAGAZINE.COM



Highly Mobile Premium-Class Business Companion

TUXEDO InfinityBook Pro 15 - Gen10

The TUXEDO InfinityBook Pro 15 – Gen10 is a highly mobile premium-class business companion, available either with a powerful AMD Ryzen AI 300-series processor or an efficient Intel Core Ultra H processor. Its elegant full-aluminum chassis features a 15.3″ display (2,560 × 1,600 px, 500 cd/m², 100% sRGB) in a compact format (34.2 × 24.0 cm, only approx. 1.75 kg), equipped with a 99 Wh battery offering up to approx. 10–11 hours of runtime, up to 128 GB DDR5 RAM, and up to 8 TB PCIe SSD storage — designed for maximum performance in everyday mobile use. Whether for creative professionals or business experts, it delivers uncompromising quality and flexibility.

Linux compatible

Up to 5 Years Guarantee

Immediately ready for use

TUXEDO
tuxe.do/lxadmin90

Made in Germany

German Data Privacy

German Tech Support

Happy Anniversary to Us

Ken looks back at 15 years of ADMIN

It's difficult to believe that this year is the 15th anniversay of *ADMIN*. I've written for almost every issue and am pleased to be involved. I still remember in 2010 when Editor in Chief Joe Casad first contacted me about the new magazine that was to "focus on system administration," which he knew was smack dab in the center of my wheelhouse. Joe knew me from writing for other publications under his editorial control and thought I'd be a great fit. He was right, and the rest is history – 15 years of history.

Also of note is that *Linux Magazine* also reached its 25th anniversary this year. It's been a great run of publications for Linux New Media, and I congratulate Joe and the rest of the team for curating, editing, and elevating open source and technical IT content for us all. I'm proud to be part of this tech history.

I've been here since *ADMIN* launched, having written an article about OpenSolaris Zones in that first special issue, referred to as issue 0. And that's a funny, wonderful thing about technology: a 15-year-old article can still be relevant – or one that someone published a month ago can be outdated. Zones, jails, and containers, which are all implementations of the same concept, have found new relevance in contemporary IT shops. I thought that perhaps we would have evolved past containers by now. They were a hot topic 10 years ago once Docker and Podman hit the virtual pavement, but interest seemed to wane as fast as it gained focus. Now, however, even the government is moving to containers and containerizing many of its workloads as part of a movement called Zero Trust.

Alas, I'm not a fan of dockerized workloads. Podman and Docker are sold as being more secure, but they really aren't on their own. To make them secure, you must do a bit of kernel hardening and combine multiple other technologies to improve security, such as AppArmor/SELinux, syscall filtering, namespaces, cgroups, and more. It's applying a pound of prevention for an ounce of cure compared with the relative ease of implementation of BSD jails and zones or standard `chroot` containers. As I've written in this Welcome column before, however, when one IT shop does something and gets some notice for it, everyone else jumps into the fray as in the myth of lemmings jumping off a cliff.

You see? In a decade and a half, certain technology solutions haven't evolved much. But, I digress. My point is that *ADMIN* has provided you with excellent technology information for an incredible 15 years and will continue to do so. Even as a seasoned system administrator and cross-platform know-it-all, I've spent countless hours with the magazine next to me at my desk (yes, I love the print version) while I enter code, try commands, and install packages hoping to replicate the often highly technical and always informative solutions from our contributors.

The success of *ADMIN* can be attributed to its gap-filling presence. The writers are all practicing technology professionals – which makes a big difference to those of us who are in the trenches – supporting multiple applications, distributions, versions, and environments. Being a system administrator isn't for the faint of heart, but our contributors' articles can take some of the sting and frustration out of it. That's our mission and it's stated succinctly on the [admin-magazine.com] site: "Founded in 2010, *ADMIN* is the source for technical solutions to the real-world problems sys admins face every day."

I'd like to thank all of those responsible over the years that have made this anniversary possible: Brian Osborn, Publisher and Linux New Media CEO; Joe Casad, Editor in Chief; Rita L Sooby, Managing Editor; Dena Friesen, Layout; Lori White, Managing Editor, Layout, and Ken Hess Wrangler; Amber Ankerholz, News Editor; Ian Travis, Localization and Translation; and Amy Pettle and Aubrey Vaughn, Copy Editors. Without this team, there would be no anniversaries. Without the great writers and contributors, there would be no content. And without you, the loyal system administrator, reader, and subscriber, there would be no need to publish.

Ken Hess • Senior ADMIN Editor

Lead Image © antoinepham2903, 123RF.com

ADMIN
Network & Security

Features

10 **Checking VoIP Networks**
Open source tools can help you discover vulnerabilities in VoIP networks, fix typical problems, and boost infrastructure security and functionality.

Service

3 **Welcome**
6 **News**
96 **Back Issues**
97 **Call for Papers**
98 **Coming Next Month**

@adminmag

ADMIN magazine

@admin-magazine.com

@adminmagazine@hachyderm.io

Tools

16 **Microsoft Loop**
This flexible collaboration tool aims to redefine seamless integration and dynamic teamwork for improved real-time collaboration.

20 **Simulating Network Traffic**
Network simulators provide the information admins need to plan and grow wired and wireless networks.

26 **Ollama+Cursor**
An on-premises open source AI developer workflow uses Ollama+Cursor for maximum data sovereignty and control.

34 **Sigstore Cosign**
The Sigstore Cosign, Fulcio, and Rekor ecosystem can improve software supply chain security by making signatures easy to use and difficult to subvert.

38 **bWatch**
After 21 years, our HPC author returns to bWatch, a simple cluster monitoring tool from the early days of Beowulf.

Containers and Virtualization

42 **Kata Containers**
We explore the architecture and configuration of Kata Containers on Ubuntu with Docker, focusing on how lightweight virtual machines can deliver strong workload isolation while preserving the agility and performance of traditional containers.

Security

46 **Administrator Protection**
Microsoft's replacement for the deprecated User Account Control security mechanism provides state-of-the-art protection by employing the principle of least privilege and just-in-time privilege assignment.

50 **Tetragon**
This comprehensive security suite for Kubernetes is based on eBPF and focuses on monitoring and proactive threat detection with tracing for popular programming languages.

Code and listings for *ADMIN* can be found here:
https://linuxnewmedia.thegood.cloud/s/9nFQcFb2p8oRMEJ

10 | VoIP Network Security

Discover vulnerabilities and boost infrastructure security

Session Initiation Protocol provides both the underpinnings for VoIP and a potential attack vector for hackers. Open source tools can help you test and secure your VoIP networks.

Highlights

26 Ollama+Cursor
The on-premises privacy-first stack of Ollama, for local LLM serving, with the Cursor AI-native IDE prioritizes data sovereignty, cost control, and developer agility.

42 Kata Containers
Blend virtualization with containerization to provide strong workload isolation and an important layer of security in containerized environments.

87 MLOps Pipeline
Data engineering, model training, containerization, and Kubernetes-based deployment are seamlessly integrated into an end-to-end MLOps pipeline.

Management

56 Secure AIX
Modernizing AIX security with multifactor authentication has never been easier than with Google Authenticator, which brings two-factor authentication to your AIX system.

64 Puppet Autosign Policies
Automated Puppet enrollment on Debian workstations can be enforced with autosign scripts that implement your security policy with a one-time enrollment script and an automated installation preseed.

Nuts and Bolts

72 x2openEuler Migration
Easily migrate your old CentOS 7 systems to the modern openEuler with the x2openEuler migration toolset.

78 Logger
Store user-specified messages in syslog to gather information for HPC admin tasks.

82 Hotpatching Win Server 2025
Hotpatching technology on Windows Server 2025 reduces downtime and maintenance windows, but it comes at a price in the truest sense of the word.

87 MLOps Pipeline
A production-grade MLOps pipeline automates the entire machine learning lifecycle, from data ingestion to deployment in a Kubernetes environment.

On the DVD

Ubuntu 25.10 Server

The "Questing Quokka" server image permanently installs Ubuntu on a computer minus a graphical user interface. The release "comes with compatibility enhancements at the silicon level, accessibility upgrades, and a robust security posture that sets the stage for the next LTS." [https://canonical.com/blog/canonical-releases-ubuntu-25-10-questing-quokka]

Version 25.10 comes with memory-safe implementations of coreutils and sudo-rs, improvements in trusted platform module (TPM)-backed full disk encryption, and support for nested virtualization on Arm.

ubuntu
25.10 Server
ISSUE 90/2025
ADMIN
Network & Security
DVD

News for Admins

Tech News

Routers Remain Top Target for IoT Attacks, Per Zscaler Report

Routers reimain the most targeted IoT devices, accounting for more than 75 percent of all attacks, according to the Zscaler ThreatLabz 2025 Mobile, IoT, and OT Threat Report (https://www.zscaler.com/campaign/threatlabz-mobile-iot-ot-report).

Attackers exploit various router vulnerabilities to execute commands, propagate malware, and expand botnets, the report states. Additionally, it notes that “Netgear routers are an especially popular target.” Common exploitation methods include command injection and directory traversal techniques that “often exploit unauthenticated remote code execution (RCE) vulnerabilities, such as CVE-2016-10174 (https://nvd.nist.gov/vuln/detail/cve-2016-10174) and CVE-2018-10561 (https://nvd.nist.gov/vuln/detail/cve-2018-10561) [which] allow threat actors to bypass authentication and execute scripts remotely.”

Top IoT threat findings include:

- The US is the top target for IoT attacks, with 54.1 percent of activity.
- Routers account for more than 75 percent of all attacks, which are mainly driven by command injection vulnerabilities.
- Manufacturing and transportation sectors together are the target of 40 percent of all IoT malware attacks.
- The majority of IoT malware is linked to the Mirai and Mozi malware families, with roughly 40 percent of blocked transactions linked to the Mirai family alone.

Mobile threats are also on the rise, with Android malware transactions increasing by 67 percent year-over-year. According to the report, this trend is “fueled by spyware and banking malware targeting trusted marketplaces and hybrid work environments.” For example, the ThreatLabz team identified 239 malicious Android applications, which were collectively downloaded 42 million times on the Google Play Store.

Read the complete report at Zscaler, Inc (https://connect.notified.com/Tracker?data=bH8mjBIWYgGI1Y6dcrzJsHcviNeIgV-2g9NycMIWWjEtITQJJAsvi2-fB2OVasntGpvuOMI8P25FTy7VT8yYoeDS9xDRQcjKM9FKYBAXCo4=000000000000).

Palo Alto Networks Launches Prisma AIRS 2.0

Palo Alto Networks has announced Prisma AIRS 2.0 (https://www.deploybravely.com/), a major platform upgrade that provides “end-to-end protection across the entire AI application lifecycle.”

Prisma AIRS 2.0 offers “a clear view of your entire AI ecosystem, helps you assess emerging risks, and defends your organization against threats unique to AI,” says Ian Swanson (https://www.paloaltonetworks.com/blog/2025/10/prisma-airs-powering-secure-ai-innovation/) in the announcement blog post.

The platform, which is “continuously validated by autonomous AI red teaming,” secures interactions between AI models, agents, data, and users through the following features:

- AI model security: Analyzes the model itself to detect threats like architectural backdoors, data poisoning, and malicious code hidden within the model layers.
- Comprehensive discovery: Finds all AI agents in use across your environments, including shadow or inactive agents that still hold access.

Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update
bit.ly/ADMIN-Update

Lead Image © vlastas, 123RF.com

- Permission and configuration management: Detects misconfigurations, shared credentials, and over-permissive identities that violate least-privilege policies.
- Continuous hardening: Checks for configuration drift and risky trust chains between agents and third-party tools.

Learn more at Palo Alto Networks (https://www.paloaltonetworks.com/).

Fedora 43 Released

The Fedora team has announced the release of Fedora 43 (https://fedoramagazine.org/announcing-fedora-linux-43/).

Notable changes, according to Fedora project leader Jef Spaleta, are:

- Anaconda WebUI (https://fedoramagazine.org/anaconda-installer-redesign/) is now the default installer interface for Fedora Linux 43 Spins (https://www.fedoraproject.org/spins/).
- GNOME is now Wayland-only. “GNOME upstream has deprecated X11 support, and has disabled it as a compile time default in GNOME 49. Upstream GNOME plans to fully remove X11 support in GNOME 50,” Spaleta says.
- Fedora Linux 43 is the first release featuring RPM 6.0 (https://rpm.org/releases/6.0.0). “RPM 6.0 provides some interesting security enhancements, like multiple key signing of packages. This should help future-proof package signing as we transition to post-quantum-crypto OpenPGP (https://www.ietf.org/archive/id/draft-wussler-openpgp-pqc-01.html) keys in future releases,” Spaleta notes.

Additionally, as of Fedora 43, Fedora CoreOS updates will be delivered exclusively as OCI images, instead of using an OSTree repository.

See the project release notes (https://docs.fedoraproject.org/en-US/fedora/latest/release-notes/) as well as the “what’s new” articles for Fedora Workstation 43 (https://docs.fedoraproject.org/en-US/fedora/latest/release-notes/), Fedora KDE Plasma Desktop (https://fedoramagazine.org/whats-new-in-fedora-kde-plasma-desktop-43/), and Fedora Atomic Desktops (https://fedoramagazine.org/whats-new-fedora-atomic-desktops-in-fedora-linux-43/) for more details.

Zorin OS 18 Now Available

The Zorin Group has officially launched Zorin OS 18 (https://zorin.com/os/download/), a major release of the popular desktop operating system.

This latest release of Zorin OS, which is touted in the announcement (https://blog.zorin.com/2025/10/14/zorin-os-18-has-arrived/) as “the perfect Windows replacement for everyday users, professionals, and everyone in between,” comes at an opportune time as Windows 10 reached its end of life earlier this month.

New features and updates in Zorin OS 18 include:

- Redesigned desktop with three new layouts and new theme colors
- New default window tiling manager
- Built-in Web Apps tool
- OneDrive file integration
- More pre-installed apps for the Pro and Education editions

Zorin OS 18 offers a “fast, user-friendly, and secure experience,” the company says, to help users quickly and easily transition away from Windows.

Learn more at Zorin OS (https://zorin.com/).

Linux Magazine Celebrates 300th Issue

The Linux Magazine (https://www.linux-magazine.com/) team recently published the 300th issue (https://www.linux-magazine.com/Issues/2025/300) of the magazine, marking 25 years of covering Linux, open source software, and a world of related topics.

“A quarter century is a long time for any magazine, but for a computer magazine, we are in rare company. I just want to give a shout-out to our great team, our great readers, and our great topic: Linux and the open source community,” says Editor in Chief Joe Casad.

Throughout the past 25 years, the magazine has provided in-depth coverage and technical expertise for a rapidly changing array of hardware- and software-related innovations and approaches, such as:

- Open source development
- Clusters
- High-performance computing
- Cloud computing
- Containers
- Machine learning and AI
- Security and privacy

The magazine also reports on new tools, tracks the development of various Linux distributions, and offers tips and best practices for dealing with modern computing scenarios.

"Linux keeps evolving, which means that we get to keep evolving with it, and we never run out of exciting new things to write about. … As long as Linux keeps changing, our authors and our readers will change with it, and we will chronicle it all as it unfolds," Casad says.

Learn more at *Linux Magazine* online (https://www.linux-magazine.com/Issues/2025/300/maddog-s-Doghouse).

Tails 7.0 Announced

The Tails team has released version 7.0 of Tails (https://tails.net/news/version_7.0/), a portable Linux-based operating system that "uses the Tor network to protect your privacy online and help you avoid censorship." This release is a major update that includes new versions of many applications.

Tails 7.0, which is based on Debian 13 (Trixie) and GNOME 48 (Bengaluru), replaces GNOME terminal with GNOME Console (https://apps.gnome.org/Console/) as the default terminal emulator and features GNOME Loupe (https://apps.gnome.org/Loupe/) as the default image viewer, among other software updates.

This version also starts 10–15 seconds faster on most computers, the announcement says, and has an increased memory requirement from 2GB of RAM to 3GB.

Download the latest version and learn more at Tails (https://tails.net/news/version_7.0/).

openSUSE Leap 16 Released

The openSUSE Project has released openSUSE Leap 16 (https://get.opensuse.org/leap/16.0/).

The latest version features an updated lifecycle, a fresh software stack, a new installer (https://en.opensuse.org/Agama), a longer support cycle, and simplified migration options (https://en.opensuse.org/SDB:System_upgrade). "Unless the project makes strategic changes, annual minor releases are expected to continue until 2031 with the release of Leap 16.6. A successor to Leap 16 is expected in 2032," the announcement says (https://news.opensuse.org/2025/10/01/next-chapter-opens-with-leap-release/).

Leap 16 also includes new hardware requirements, namely: "Leap 16 now requires x86-64-v2 as a minimum CPU architecture level, which generally means CPUs bought in 2008 or later."

"With a newly extended support period of 24 months, Leap gives users, developers, and sysadmins one of the best and most reliable Linux experiences available," the announcement states.

Learn more at openSUSE (https://news.opensuse.org/2025/10/01/next-chapter-opens-with-leap-release/).

PostgreSQL 18 Released

The PostgreSQL team has released version 18 of the open source database.

According to the announcement, PostgreSQL 18 (https://www.postgresql.org/docs/18/release-18.html) offers improved performance through "a new I/O subsystem that has demonstrated up to 3 × performance improvements when reading from storage." This new asynchronous I/O subsystem "lets PostgreSQL issue multiple I/O requests concurrently instead of waiting for each to finish in sequence."

Other PostgreSQL 18 features include:

- Faster upgrade times
- Improved text processing
- A uuidv7() function for improved indexing and read performance for UUIDs
- Easier integration with single-sign on (SSO) systems with support for OAuth 2.0 authentication
- Accelerated query performance

Learn more at PostgreSQL (https://www.postgresql.org/about/news/postgresql-18-released-3142/).

Alpha Version of KDE Linux Released

The KDE team has announced the alpha release of its KDE Linux operating system and invites users to test it.

According to the team announcement (https://pointieststick.com/2025/09/06/announcing-the-alpha-release-of-kde-linux/), KDE Linux "is intended for daily driving that showcases Plasma and KDE software in the best light, and makes use of modern technologies."

The Testing Edition provides built-from-git-master versions of Plasma, KWin, and KDE apps, the website says (https://kde.org/linux/), and is "designed to be safe, maintainable, functional, and modern." However, the alpha release contains multiple known issues (https://community.kde.org/KDE_Linux#Current_state) and has so far received only limited testing.

That said, things that work include:

- Getting apps from Flathub
- Developing KDE and third-party software
- System updating using Discover and the command-line
- System rollbacks

KDE Linux Alpha is not recommended for production but should be generally "usable for adventurous people."

Learn more at KDE (https://kde.org/linux/).

Together AI Announces Instant Clusters

Together AI announced the general availability of its Together Instant Clusters (https://www.together.ai/instant-gpu-clusters) service, which automates the provisioning of GPU clusters.

The "pay-as-you-go" service lets organizations quickly spin up NVIDIA GPU clusters, ranging from a single node with eight GPUs to large, multi-node systems with hundreds of processors. "Each cluster is built using NVIDIA H100 (80GB SXM) GPUs, engineered for AI training, inference, and fine-tuning at scale," the company says. Instant Clusters also feature NVIDIA Quantum-2 InfiniBand and NVLink networking, with the choice of Kubernetes or Slurm for orchestration.

According to the company, Instant Clusters can be provisioned in minutes and include the following components:

- GPU Operator to manage drivers and runtime software
- Ingress controller to handle traffic into your cluster
- NVIDIA Network Operator for high-performance networking
- Cert manager for secure certificates and HTTPS endpoints

See pricing (https://www.together.ai/pricing#instant-clusters) and other details at Together AI (https://www.together.ai/).

Microsoft's 6502 BASIC is Now Open Source

Microsoft has officially released its 6502 BASIC programming language under the open source MIT license.

"Microsoft BASIC began in 1975 as the company's very first product: a BASIC interpreter for the Intel 8080, written by Bill Gates and Paul Allen for the Altair 8800," says Scott Hanselman, Vice President, Developer Community at Microsoft. 6502 BASIC was later licensed by Commodore for a flat fee of $25,000. "That decision put Microsoft's BASIC at the heart of Commodore's machines and helped millions of new programmers learn to code."

This newly open sourced version is "BASIC M6502 8K VER 1.1, the 6502 BASIC lineage that powered an era of home computing and formed the foundation of Commodore BASIC in the PET, VIC-20, and the legendary Commodore 64," Hanselman says.

According to the project's GitHub page (https://github.com/microsoft/BASIC-M6502), "this assembly language source code represents one of the most historically significant pieces of software from the early personal computer era."

Learn more at Microsoft (https://opensource.microsoft.com/blog/2025/09/03/microsoft-open-source-historic-6502-basic/).

Check your VoIP networks with SIPVicious and sngrep

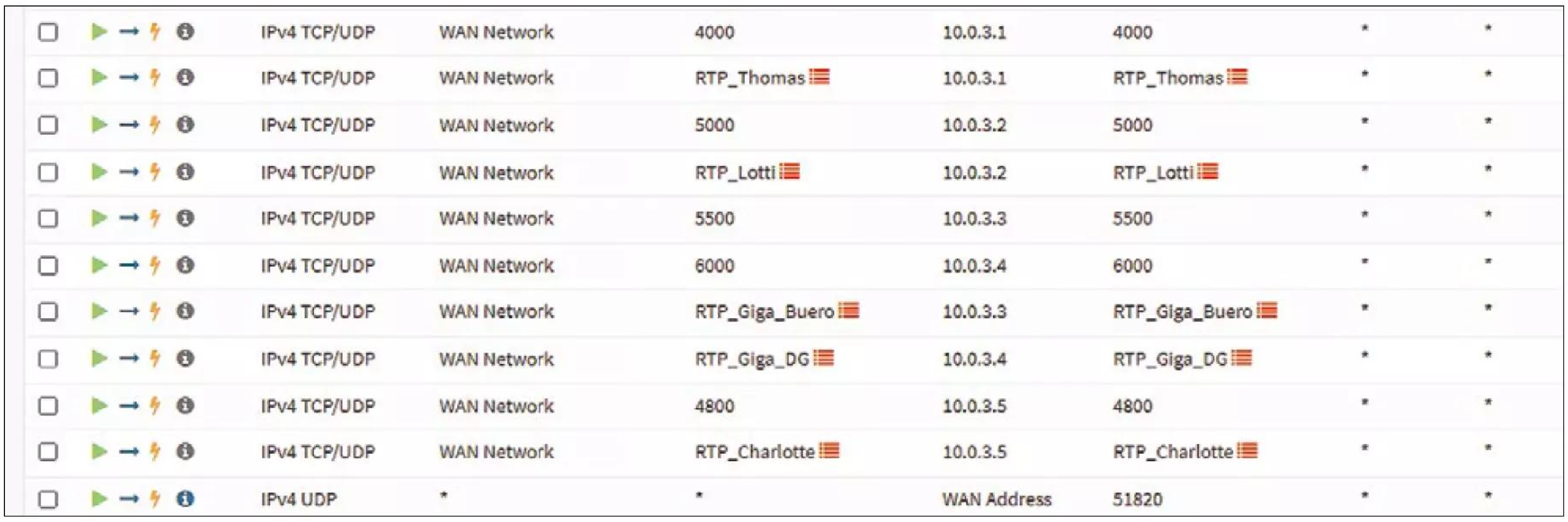
Free Line

Open source tools can help you discover vulnerabilities in VoIP networks, fix typical problems, and boost infrastructure security and functionality. By Thomas Joos

In the corporate environment, telecommunications are now almost entirely based on VoIP, the key component being the Session Initiation Protocol (SIP), which provides both the underpinnings for calls and a potential attack vector for hackers. The most common pitfalls include insecure passwords, incorrectly configured servers, and blocked ports. In this article, I introduce open source tools such as SIPVicious, `sngrep`, and Wireshark, which you can use to test and secure your VoIP networks.

Despite email, instant messaging, and other forms of communication, a large part of corporate communication is still reliant on phone calls. The VoIP technology begins with a call. A terminal device is then registered with the SIP server by the `REGISTER` message. The terminal device sends its SIP ID and access data to the server, which can be a local telephone system or a cloud-based system to which the local terminal devices connect. The connection is then established by the `INVITE` method.

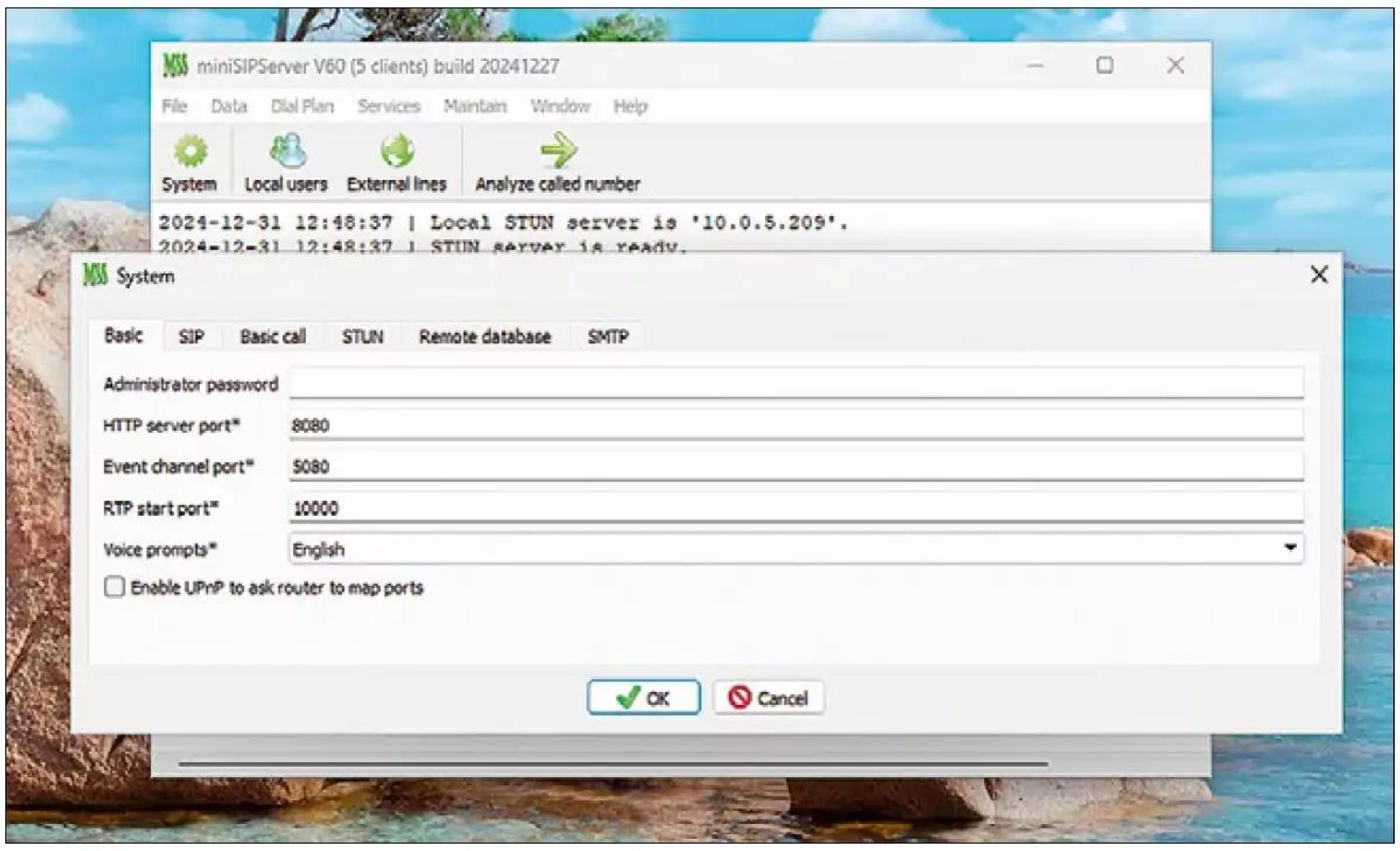
The SIP server signals the incoming invitation to the target device and uses the Session Description Protocol (SDP) to negotiate the connection parameters, such as the codec and ports for the media stream (transmitted with the Real-Time Transport Protocol, RTP). The data transfer relies on RTP, whereas control tasks are handled by SIP (Figure 1). Finally, the connection is terminated with the `BYE` command. Problems often arise from blocking by firewalls that filter out SIP or RTP packets, or because of incorrect configurations on the terminal device and on the server side. The tools described in this article can help you troubleshoot these issues.



☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	4000	10.0.3.1	4000	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	RTP_Thomas ☰	10.0.3.1	RTP_Thomas ☰	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	5000	10.0.3.2	5000	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	RTP_Lotti ☰	10.0.3.2	RTP_Lotti ☰	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	5500	10.0.3.3	5500	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	6000	10.0.3.4	6000	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	RTP_Giga_Buero ☰	10.0.3.3	RTP_Giga_Buero ☰	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	RTP_Giga_DG ☰	10.0.3.4	RTP_Giga_DG ☰	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	4800	10.0.3.5	4800	*	*
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	WAN Network	RTP_Charlotte ☰	10.0.3.5	RTP_Charlotte ☰	*	*
☐	▶ → ⚡ ℹ	IPv4 UDP	*	*	WAN Address	51820	*	*

Figure 1: Firewall rules can affect voice quality and whether VoIP devices can connect. Tools help you troubleshoot.

Photo by Quino Al on Unsplash



```
2024-12-31 12:48:37 | Local STUN server is '10.0.5.209'.
2024-12-31 12:48:37 | STUN server is ready.
```

Administrator password:
HTTP server port*: 8080
Event channel port*: 5080
RTP start port*: 10000
Voice prompts*: English
☐ Enable UPnP to ask router to map ports

Figure 2: The ports used on the SIP server play an important role in communication, especially with regard to allowed ports on firewalls and, in some cases, on network switches.

Important Ports

By default, SIP uses port 5060 for unencrypted connections and port 5061 for encrypted connections with help from the Transport Layer Security (TLS) protocol, although the ports can be freely selected. RTP is used to transmit media streams such as voice or video; it typically uses ports in the range of 10000 to 20000. Again, these ports are not fixed and vary depending on the configuration.

If the SIP port is blocked, end devices cannot register with the SIP server, which means they are unable to initiate or receive calls. On the other hand, if the firewall blocks RTP ports, connections are interrupted or audio is missing in the calls because the media streams do not reach their destination.

Tools such as `sngrep` can test whether `REGISTER` or `INVITE` packets are reaching the SIP server. If these packets are not arriving, a firewall is often the cause. You can then specifically allow the ports in question on the firewall or set up port forwarding. More in-depth analysis with Wireshark helps identify the exact location of the block by tracking the packets en route from the client to the server, which also ensures that the firewall rules only allow access for authorized devices (Figure 2).

SIP Extensions: Risks and Solutions

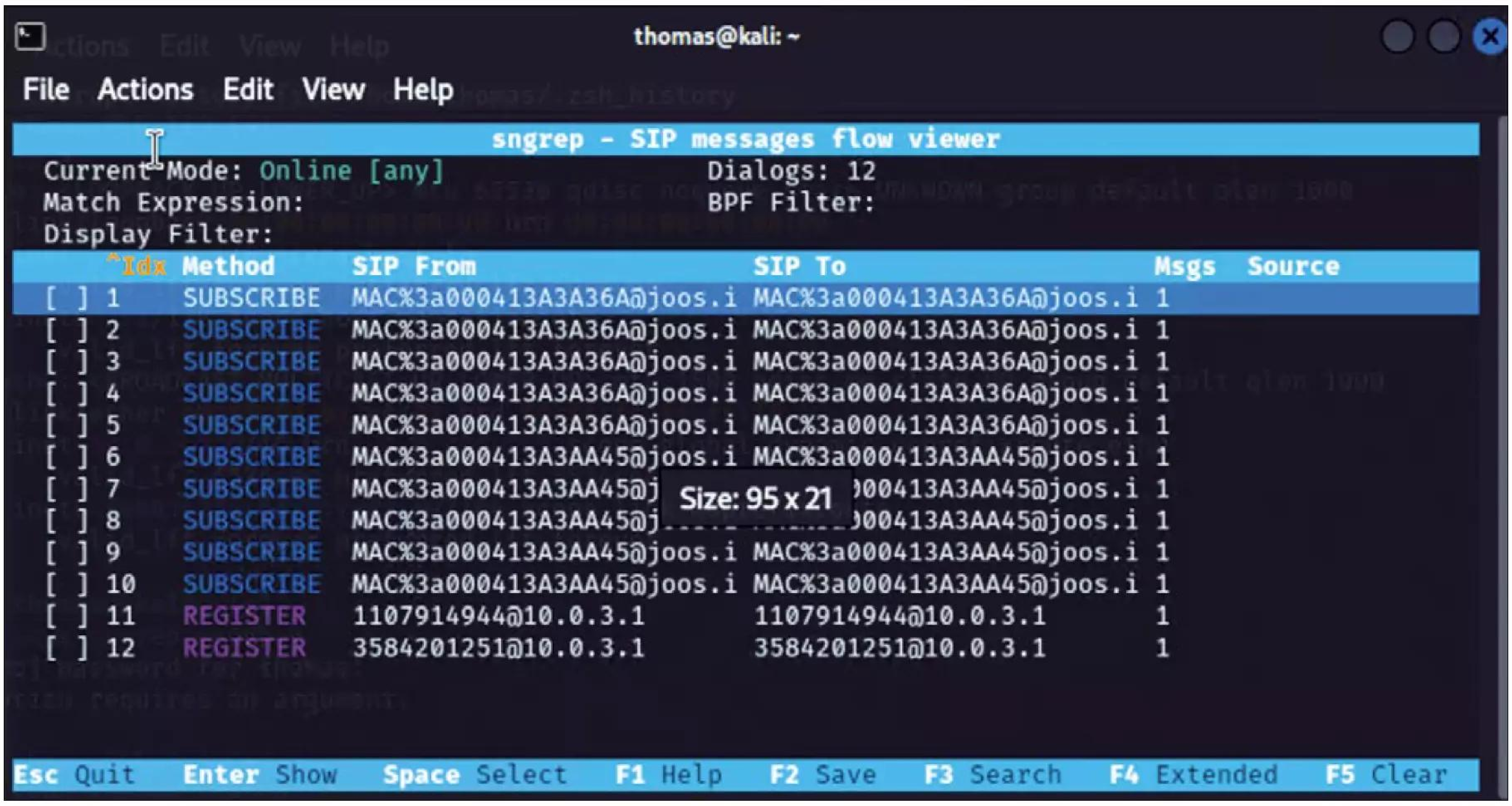
SIP extensions are the identities in a VoIP system. Each extension represents a specific number or terminal device connected to a SIP server.

The extensions need to be registered and authenticated to prevent unauthorized access or attacks. Typical problems include incorrect configurations, weak passwords, or inadequate authentication mechanisms. These vulnerabilities can prompt attacks such as brute forcing, spoofing, or unauthorized use.

Tools such as SIPVicious can be used to check which extensions are active on your server and whether they require authentication. The command

```
svwar 192.168.1.103
```

shows which extensions exist for the IP address under investigation and



```
                          sngrep - SIP messages flow viewer
 Current Mode: Online [any]                      Dialogs: 12
 Match Expression:                               BPF Filter:
 Display Filter:
      ^Idx Method     SIP From                 SIP To                   Msgs  Source
 [ ] 1    SUBSCRIBE  MAC%3a000413A3A36A@joos.i MAC%3a000413A3A36A@joos.i 1
 [ ] 2    SUBSCRIBE  MAC%3a000413A3A36A@joos.i MAC%3a000413A3A36A@joos.i 1
 [ ] 3    SUBSCRIBE  MAC%3a000413A3A36A@joos.i MAC%3a000413A3A36A@joos.i 1
 [ ] 4    SUBSCRIBE  MAC%3a000413A3A36A@joos.i MAC%3a000413A3A36A@joos.i 1
 [ ] 5    SUBSCRIBE  MAC%3a000413A3A36A@joos.i MAC%3a000413A3A36A@joos.i 1
 [ ] 6    SUBSCRIBE  MAC%3a000413A3AA45@joos.i MAC%3a000413A3AA45@joos.i 1
 [ ] 7    SUBSCRIBE  MAC%3a000413A3AA45@j  Size: 95 x 21  00413A3AA45@joos.i 1
 [ ] 8    SUBSCRIBE  MAC%3a000413A3AA45@j                 00413A3AA45@joos.i 1
 [ ] 9    SUBSCRIBE  MAC%3a000413A3AA45@joos.i MAC%3a000413A3AA45@joos.i 1
 [ ] 10   SUBSCRIBE  MAC%3a000413A3AA45@joos.i MAC%3a000413A3AA45@joos.i 1
 [ ] 11   REGISTER   1107914944@10.0.3.1       1107914944@10.0.3.1       1
 [ ] 12   REGISTER   3584201251@10.0.3.1       3584201251@10.0.3.1       1

Esc Quit   Enter Show   Space Select   F1 Help   F2 Save   F3 Search   F4 Extended   F5 Clear
```

Figure 3: The `sngrep` tool helps analyze SIP traffic on the network.

whether or not they are protected. If an extension does not require authentication, you will find a note to this effect in the output.

sngrep

To facilitate troubleshooting, the `sngrep` [1] tool visualizes SIP communication by displaying SIP messages and their flow in a clear-cut view. After typing

```
apt install sngrep
```

to install `sngrep` on Linux, launch the tool with admin privileges. After launch, `sngrep` lists all the SIP packets flowing through the network, showing the individual steps of the SIP traffic so that you can home in on issues quickly.

For example, you can check whether registration requests are successful or devices are receiving *401 Unauthorized* messages from the server. Successful registrations can also be seen. On top of this, you can use the tool to check whether the device is sending `REGISTER` packets (Figure 3). If the packets are reaching their target, you can then analyze the server response. A server that remains silent indicates a network problem, such as blocking by a firewall.

If `sngrep` does not display all the SIP messages, or some are missing, potential causes include invisible SIP traffic, incorrect selection of the network interface, or missing permissions. Your first step is to check whether SIP traffic exists

```
sudo tcpdump -i eth0 port 5060
```

If you use different ports for SIP, you will need to modify the command. Alternatively, filter the traffic in Wireshark with a filter, such as:

```
sip || udp.port == 5060
```

Before you use `sngrep`, you need to select the correct network interface. Another potential reason for information being missing is encrypted SIP traffic. SIP messages over TLS typically use port 5061. You can use the

```
sudo sngrep -p 5061
```

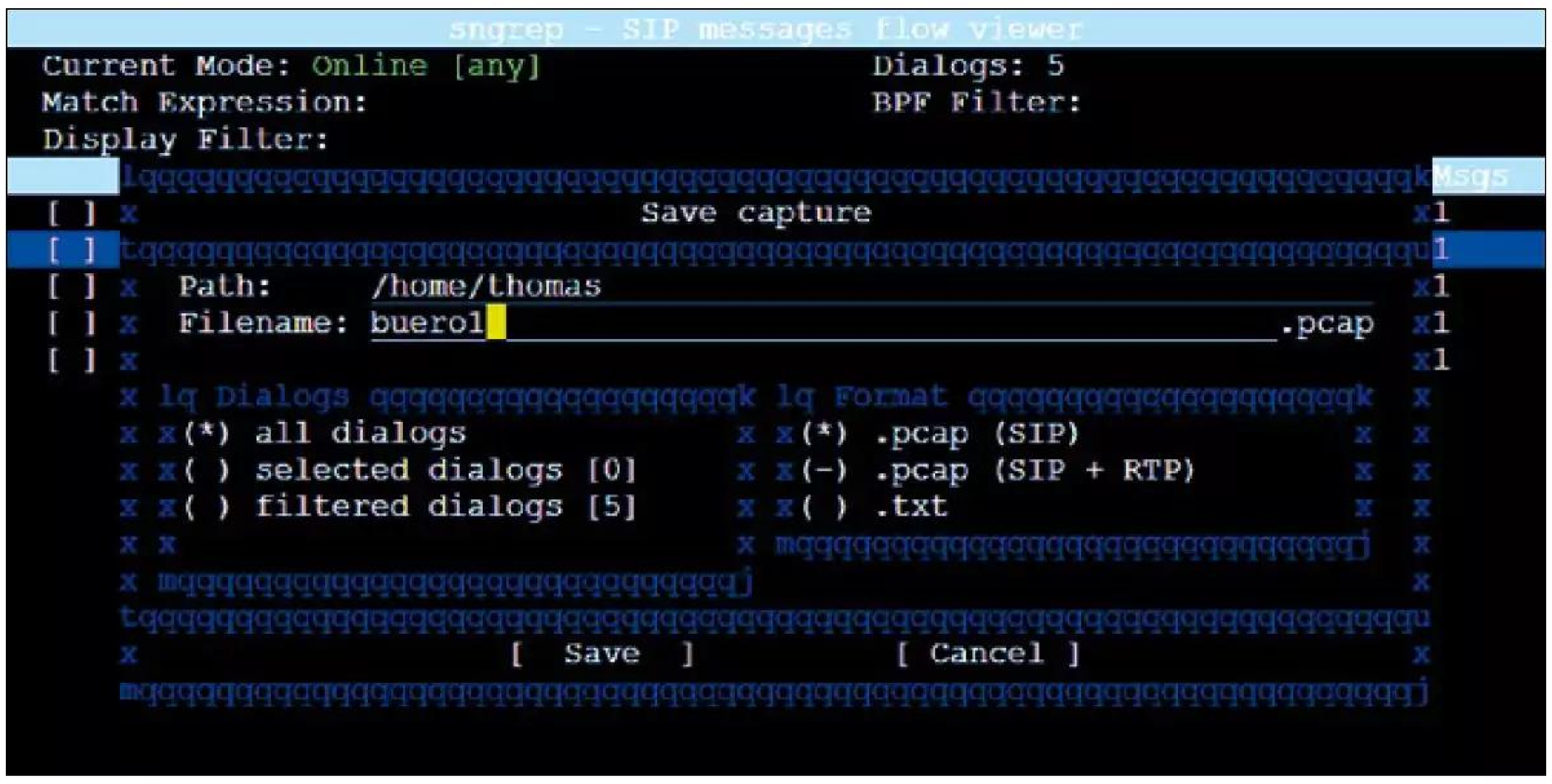
command to analyze the traffic on this port. Additionally, you can check whether firewalls or NAT configurations are blocking or redirecting SIP packets. The

```
sudo iptables -L
```

command shows you whether the ports in question are staying open. As soon as `sngrep` detects SIP traffic, the individual packets are shown in the window. When you call up the details of a packet, you will find the IP address of the end device, usually also the type, plus further information about the packet, such as successful communication with the server. At this point, it is interesting to save the data in a PCAP file (Figure 4), which you can then analyze in-depth with Wireshark.

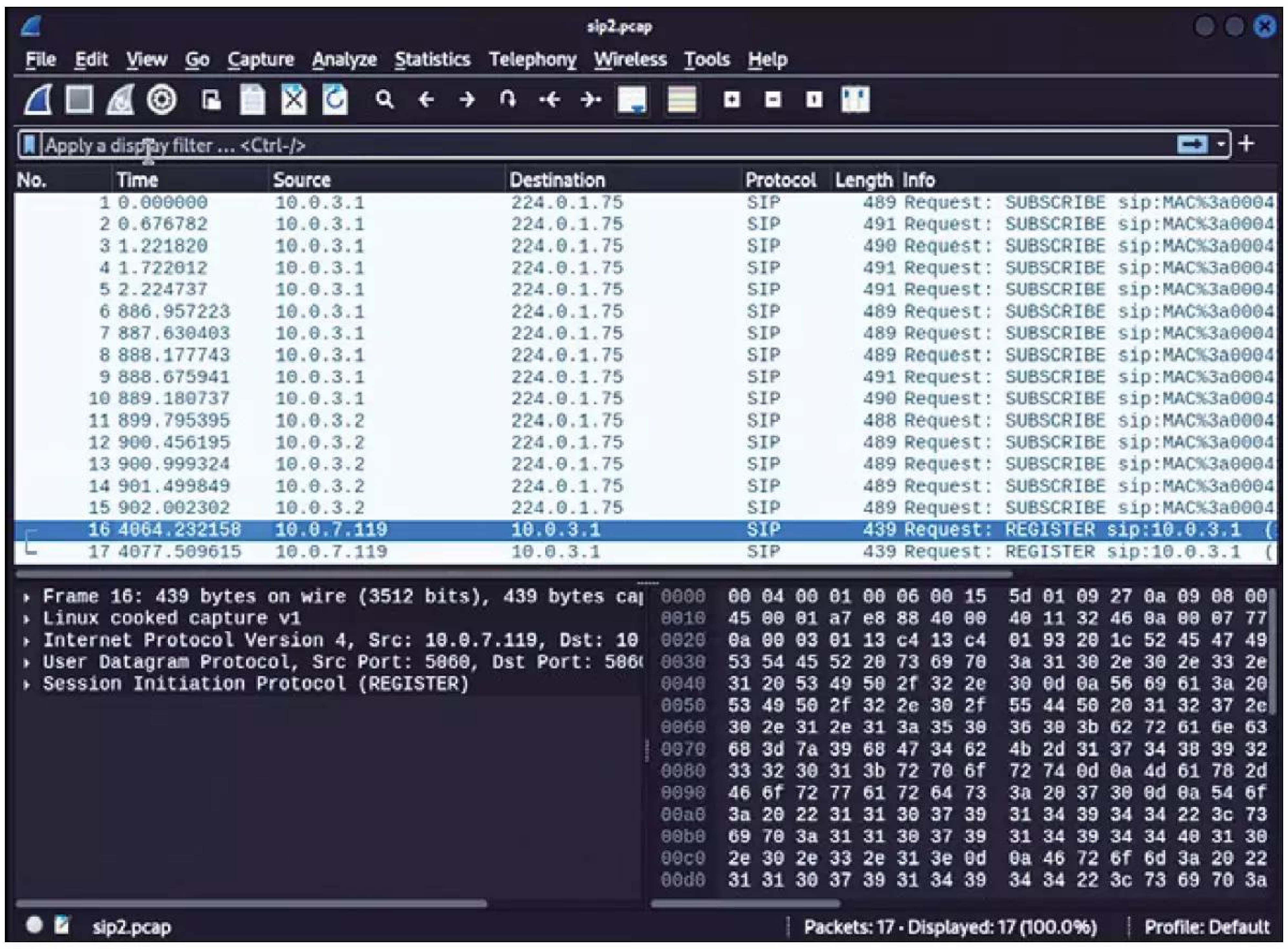
Protocol Analysis with Wireshark

Wireshark [2] complements `sngrep` by providing detailed insights into the structure of SIP and RTP packets. For example, you can use `sngrep` to capture a faulty registration, export the data to a PCAP file, and use Wireshark to examine the SIP headers to identify incorrect parameters or routing errors.



```
                  sngrep - SIP messages flow viewer
 Current Mode: Online [any]                  Dialogs: 5
 Match Expression:                           BPF Filter:
 Display Filter:
        lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqkMsgs
  [ ]   x                      Save capture                          x1
  [ ]   tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqu1
  [ ]   x  Path:      /home/thomas                                   x1
  [ ]   x  Filename:  buero1                               .pcap     x1
  [ ]   x                                                            x1
        x lq Dialogs qqqqqqqqqqqqqqqqqqqk lq Format qqqqqqqqqqqqqqqqqqqqqk x
        x x(*) all dialogs                x x(*) .pcap (SIP)             x x
        x x( ) selected dialogs [0]       x x(-) .pcap (SIP + RTP)       x x
        x x( ) filtered dialogs [5]       x x( ) .txt                    x x
        x x                               x mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj x
        x mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj                                 x
        tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqu
        x                [  Save  ]            [ Cancel ]                  x
        mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

Figure 4: You can save the analyzed data packets from `sngrep` as a PCAP file and open this file in Wireshark.



sip2.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time          Source       Destination   Protocol Length Info
    1 0.000000      10.0.3.1     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
    2 0.676782      10.0.3.1     224.0.1.75    SIP      491 Request: SUBSCRIBE sip:MAC%3a0004
    3 1.221820      10.0.3.1     224.0.1.75    SIP      490 Request: SUBSCRIBE sip:MAC%3a0004
    4 1.722012      10.0.3.1     224.0.1.75    SIP      491 Request: SUBSCRIBE sip:MAC%3a0004
    5 2.224737      10.0.3.1     224.0.1.75    SIP      491 Request: SUBSCRIBE sip:MAC%3a0004
    6 886.957223    10.0.3.1     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
    7 887.630403    10.0.3.1     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
    8 888.177743    10.0.3.1     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
    9 888.675941    10.0.3.1     224.0.1.75    SIP      491 Request: SUBSCRIBE sip:MAC%3a0004
   10 889.180737    10.0.3.1     224.0.1.75    SIP      490 Request: SUBSCRIBE sip:MAC%3a0004
   11 899.795395    10.0.3.2     224.0.1.75    SIP      488 Request: SUBSCRIBE sip:MAC%3a0004
   12 900.456195    10.0.3.2     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
   13 900.999324    10.0.3.2     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
   14 901.499849    10.0.3.2     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
   15 902.002302    10.0.3.2     224.0.1.75    SIP      489 Request: SUBSCRIBE sip:MAC%3a0004
   16 4064.232158   10.0.7.119   10.0.3.1      SIP      439 Request: REGISTER sip:10.0.3.1  (
   17 4077.509615   10.0.7.119   10.0.3.1      SIP      439 Request: REGISTER sip:10.0.3.1  (

› Frame 16: 439 bytes on wire (3512 bits), 439 bytes ca   0000  00 04 00 01 00 06 00 15  5d 01 09 27 0a 09 08 00
› Linux cooked capture v1                                 0010  45 00 01 a7 e8 88 40 00  40 11 32 46 0a 00 07 77
› Internet Protocol Version 4, Src: 10.0.7.119, Dst: 10   0020  0a 00 03 01 13 c4 13 c4  01 93 20 1c 52 45 47 49
› User Datagram Protocol, Src Port: 5060, Dst Port: 506   0030  53 54 45 52 20 73 69 70  3a 31 30 2e 30 2e 33 2e
› Session Initiation Protocol (REGISTER)                  0040  31 20 53 49 50 2f 32 2e  30 0d 0a 56 69 61 3a 20
                                                          0050  53 49 50 2f 32 2e 30 2f  55 44 50 20 31 32 37 2e
                                                          0060  30 2e 31 2e 31 3a 35 30  36 30 3b 62 72 61 6e 63
                                                          0070  68 3d 7a 39 68 47 34 62  4b 2d 31 37 34 38 39 32
                                                          0080  33 32 30 31 3b 72 70 6f  72 74 0d 0a 4d 61 78 2d
                                                          0090  46 6f 72 77 61 72 64 73  3a 20 37 30 0d 0a 54 6f
                                                          00a0  3a 20 22 31 31 30 37 39  31 34 39 34 34 22 3c 73
                                                          00b0  69 70 3a 31 31 30 37 39  31 34 39 34 34 40 31 30
                                                          00c0  2e 30 2e 33 2e 31 3e 0d  0a 46 72 6f 6d 3a 20 22
                                                          00d0  31 31 30 37 39 31 34 39  34 34 22 3c 73 69 70 3a
```

Figure 5: Wireshark enables targeted analysis of SIP traffic on the network.

Wireshark helps you analyze problems that occur at the protocol level. You can record the packets yourself with Wireshark or save the `sngrep` output and open the file directly in Wireshark for further analysis (Figure 5), which helps identify problems such as faulty headers, invalid authentication data, or RTP jitter.

RTP jitter refers to fluctuations in the delay when transmitting RTP packets on a network. Ideally, RTP packets should arrive at regular intervals to ensure stable voice or video transmission. Jitter occurs when packet transit times vary (e.g., because of network congestion, different routes, or hardware delays). These fluctuations lead to audible disturbances such as dropouts or distorted audio. Jitter is usually measured in milliseconds and should not exceed 30ms on a VoIP network.

You can use filters such as `sip` or `rtp` to improve clarity when carrying out real-time monitoring. Each phase of a call generates specific packets that are clearly distinguishable in Wireshark. The `INVITE` packet (Listing 1) initiates SIP communication.

When you open the packet, you will see the lines shown in Table 1 in the Session Initiation Protocol (INVITE) section. Shortly after the INVITE, a *100 Trying* message appears, indicating that the server is processing the request. In Wireshark, it looks like Listing 2.

You can see here that the message has come from the same call ID as the `INVITE` packet. The *180 Ringing* message, which you will find in Wireshark, occurs when the destination device is signaling the call (Listing 3, top). The contact address here shows the IP of the ringing device (192.168.1.105) in this example. As soon as the call is answered, a *200 OK* message appears (Listing 3, bottom).

The SDP data shows that the target codec is PCMU and that port 6004 is used for the media stream. The RTP data transmission starts after the `ACK` packet (Listing 4).

The sequence number and timestamp help identify delays and packet loss. Problems such as missing sequences indicate packet loss, whereas inconsistent timestamps are an indication of jitter.

An INVITE packet that does not receive a response can mean blocked ports or routing issues. In this case, you can use `sngrep` to check whether the packet is reaching the server. If you do not see a *200 OK* message, the problem is often the codec. Wireshark lets you compare the SDP data on the two devices to detect incompatibilities. A test with Sipsak,

```
sipsak -s sip:1234@192.168.1.104
```

can help confirm whether the target number is working correctly. In this way, you combine Wireshark with other tools for efficient analysis and to solve SIP problems.

SIP Analysis with `sipsak`

The `sipsak` tool [3] helps analyze and check SIP servers and endpoints (Figure 6). You can install the software, along with all the other tools, on Kali Linux. With `sipsak` you can send SIP requests to check the availability of servers, discover whether extensions are working, and view server responses. This tool is a great choice for supplementing `sngrep` and Wireshark.

One common problem when using SIP is incorrect access data or blocked ports, which prevent end devices from registering; `sipsak` lets you check whether an extension is

Table 1: SIP (INVITE) Section

Line	Description
Request-Line: INVITE sip:1234@192.168.1.104 SIP/2.0	Shows that the number 1234 is being called on the server with the IP address 192.168.1.104.
From: <sip:5678@192.168.1.101>	The caller's number.
To: <sip:1234@192.168.1.104>	Specifies the destination.
Call-ID: abc123xyz@192.168.1.101	Uniquely identifies the call.
Session Description Protocol payload	Details about the session; for example, *m=audio 5004 RTP/AVP 0* shows that port 5004 is used for the RTP stream and that the PCMU (G.711) codec is suggested.

Listing 1: INVITE Packet

```
Frame 1: 774 bytes on wire (6192 bits), 774 bytes
captured (6192 bits)
Ethernet II, Src: 00:1a:2b:3c:4d:5e, Dst:
00:6f:7g:8h:9i:0j
Internet Protocol Version 4, Src: 192.168.1.101, Dst:
192.168.1.104
User Datagram Protocol, Src Port: 5060, Dst Port: 5060
Session Initiation Protocol (INVITE)
```

Listing 2: Request Processing in Wireshark

```
Session Initiation Protocol (100 Trying)
   Status Line: SIP/2.0 100 Trying
   Via: SIP/2.0/UDP 192.168.1.101:5060;
      branch=z9hG4bK776asdhds
   From: <sip:5678@192.168.1.101>; tag=1928301774
   To: <sip:1234@192.168.1.104>
   Call-ID: abc123xyz@192.168.1.101
```

Listing 3: Wireshark Messages

```
Session Initiation Protocol (180 Ringing)
   Status Line: SIP/2.0 180 Ringing
   Contact: <sip:1234@192.168.1.105>
   To: <sip:1234@192.168.1.104>; tag=87654321
   From: <sip:5678@192.168.1.101>; tag=1928301774
   Call-ID: abc123xyz@192.168.1.101
...
Session Initiation Protocol (200 OK)
   Status line: SIP/2.0 200 OK
   Contact: <sip:1234@192.168.1.105>
   To: <sip:1234@192.168.1.104>; tag=87654321
   From: <sip:5678@192.168.1.101>; tag=1928301774
   Call-ID: abc123xyz@192.168.1.101
   SDP
    m=audio 6004 RTP/AVP 0
    a=rtpmap:0 PCMU/8000
```

Listing 4: RTP Data Transmission

```
Real-Time Transport Protocol
   [Stream setup by SDP (frame 8)]
   10.. .... Version: RFC 1889 Version (2)
   ..0. .... Padding: False
   ...0 .... Extension: False
   Sequence number: 12345
   Timestamp: 16498765
   Payload type: PCMU (0)
   SSRC: 0x1234abcd (305441741)
```

configured correctly. In combination with authentication data, `sipsak` helps you check registered users. The

```
sipsak -U -s sip:100@192.168.1.103 ➘
  -u <username> -a <password>
sipsak -U -s sip:thomas@ 10.0.5.219 ➘
  -u thomas -a test
```

commands send an authenticated request and checks whether the access credentials are correct.

If registration fails, you can use `sngrep` to visualize the communication between the client and server. SIP error codes such as *401 Unauthorized* or *403 Forbidden* provide clues as to whether the problem lies with the access credentials or the server configuration, respectively. Another potential use case is sending `OPTIONS` requests to check whether a SIP server is reachable and which functions it supports. The command

```
sipsak -s sip:192.168.1.103
```

sends an `OPTIONS` request to the server with IP address 192.168.1.103. The response provides information about availability and supported SIP methods. If you want to send this request to a specific port, you can specify the port by adding `-p`, for example,

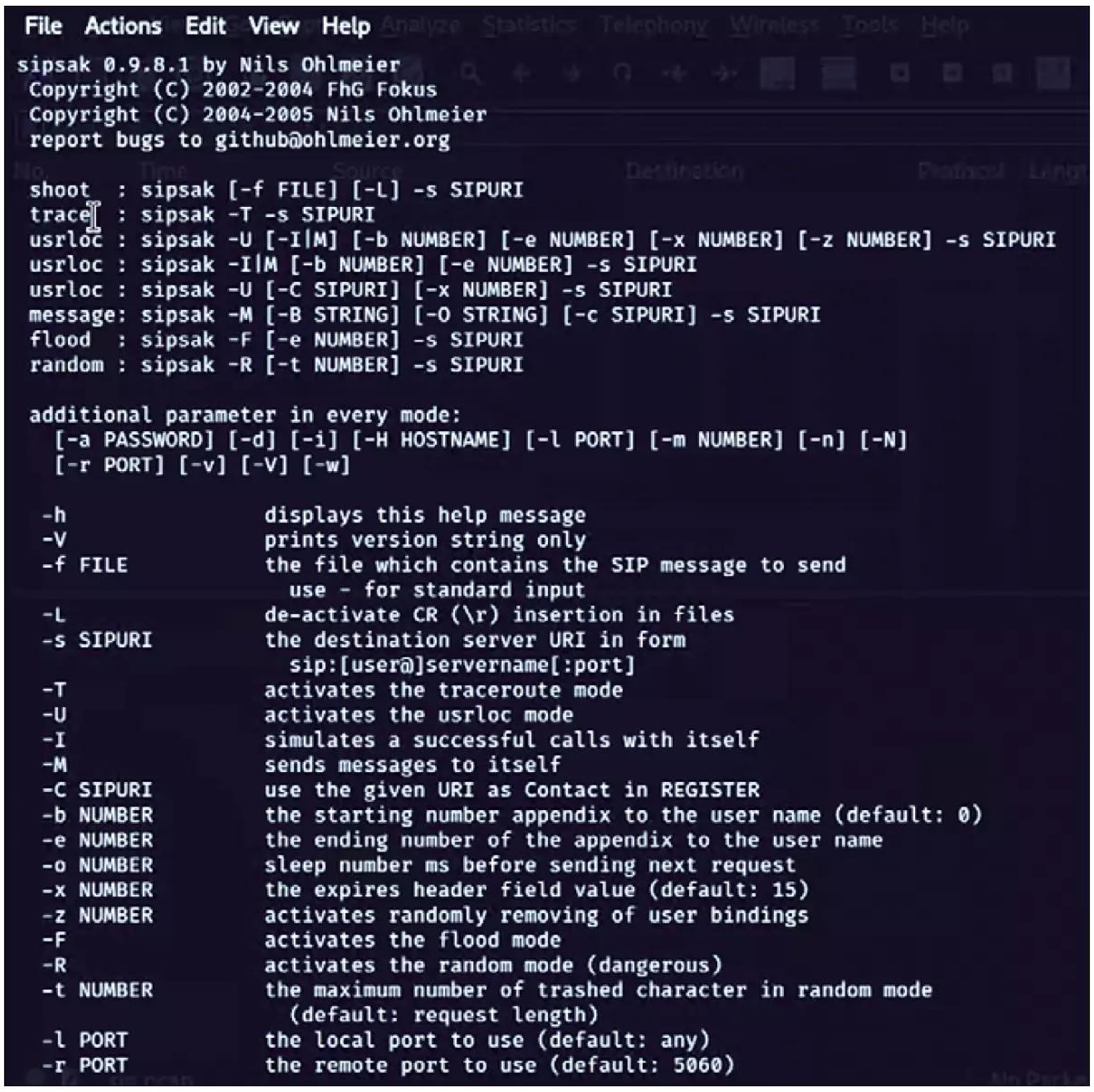
```
sipsak -s sip:192.168.1.103 -p 5061
```

if the server uses TLS-based connections on port 5061.

The use of `sipsak` combined with other tools, such as `sngrep`, Wireshark, and SIPVicious, supports more comprehensive analysis. For example, you can use `sipsak` to send a targeted request to a server and visualize the SIP messages in parallel with `sngrep`, which is useful if you want to track communication between the client and server. Whereas `sipsak` sends an `OPTIONS` request,

```
sipsak -s sip:192.168.1.103
```

`sngrep` can be used to analyze the entire message and its headers to discover potential errors.



```
sipsak 0.9.8.1 by Nils Ohlmeier
 Copyright (C) 2002-2004 FhG Fokus
 Copyright (C) 2004-2005 Nils Ohlmeier
 report bugs to github@ohlmeier.org

 shoot  : sipsak [-f FILE] [-L] -s SIPURI
 trace  : sipsak -T -s SIPURI
 usrloc : sipsak -U [-I|M] [-b NUMBER] [-e NUMBER] [-x NUMBER] [-z NUMBER] -s SIPURI
 usrloc : sipsak -I|M [-b NUMBER] [-e NUMBER] -s SIPURI
 usrloc : sipsak -U [-C SIPURI] [-x NUMBER] -s SIPURI
 message: sipsak -M [-B STRING] [-O STRING] [-c SIPURI] -s SIPURI
 flood  : sipsak -F [-e NUMBER] -s SIPURI
 random : sipsak -R [-t NUMBER] -s SIPURI

 additional parameter in every mode:
   [-a PASSWORD] [-d] [-i] [-H HOSTNAME] [-l PORT] [-m NUMBER] [-n] [-N]
   [-r PORT] [-v] [-V] [-w]

  -h                  displays this help message
  -V                  prints version string only
  -f FILE             the file which contains the SIP message to send
                        use - for standard input
  -L                  de-activate CR (\r) insertion in files
  -s SIPURI           the destination server URI in form
                        sip:[user@]servername[:port]
  -T                  activates the traceroute mode
  -U                  activates the usrloc mode
  -I                  simulates a successful calls with itself
  -M                  sends messages to itself
  -C SIPURI           use the given URI as Contact in REGISTER
  -b NUMBER           the starting number appendix to the user name (default: 0)
  -e NUMBER           the ending number of the appendix to the user name
  -o NUMBER           sleep number ms before sending next request
  -x NUMBER           the expires header field value (default: 15)
  -z NUMBER           activates randomly removing of user bindings
  -F                  activates the flood mode
  -R                  activates the random mode (dangerous)
  -t NUMBER           the maximum number of trashed character in random mode
                        (default: request length)
  -l PORT             the local port to use (default: any)
  -r PORT             the remote port to use (default: 5060)
```

Figure 6: The `sipsak` tool analyzing SIP traffic on Kali Linux.

In conjunction with Wireshark, you can optionally export the SIP messages to PCAP files for more in-depth analysis and use `sipsak` to simulate a request and simultaneously search for specific error codes *403 Forbidden* or *401 Unauthorized* in Wireshark:

```
tshark -r dump.pcap -Y sip
```

SIPVicious offers more options: When using `sipsak` to communicate with individual extensions, you can run `svwar` or `svmap` to perform parallel scans on the same server and identify further vulnerabilities, such as unsecured ports or extensions.

In another scenario, assume a user cannot log in to the SIP server; in this case, you would run the commands

```
sipsak -s sip:192.168.1.103
sipsak -U -s sip:100@192.168.1.103 ➘
  -u user -a pass
```

to make sure the server is accessible and check whether the login credentials are correct. At the same time, you could launch `sngrep` to analyze the response from the server. If the request is not reaching the server, you can use `tcpdump` to check the route the packets take across the network:

```
cpdump -i eth0 port 5060
```

Finally, you would use Wireshark to analyze the captured data packets to identify firewall or network problems.

SIPVicious

SIPVicious [4] is another open source toolset for security analysis of SIP-based systems and their configurations. It comprises several modules that perform specific tasks. The software offers extensive options for identifying SIP servers on your network and revealing their vulnerabilities, making the tool the perfect complement to `sngrep`, `sipsak`, and Wireshark. The

```
svmap 10.0.0.0/16
```

command searches through an entire subnet for SIP servers and displays their IP addresses and ports. If you specifically want to search for servers that do not use the default port (5060), use the `-p` option to specify a port list,

```
svmap 192.168.1.0/24 -p 5060,5061,5070
```

to give you an overview of the active ports and the devices responding to them.

You can also use `svmap` to scan specific devices only, such as individual servers. The commands

```
svmap 192.168.1.103
svmap 192.168.1.103 -p 5060-5080
```

let you limit the analysis to a single IP address and extend the scanning across multiple ports, to check ports that are outside the standard range.

If you use multiple segments on a network, `svmap` can be used to check specific virtual local area networks (VLANs) or subnets. By combining it with tools such as Nmap, you can identify potentially suspicious devices and use `svmap` to scan specific SIP services. For example,

```
nmap -p 5060 192.168.2.0/24 | ↩
  grep open | ↩
  cut -d ' ' -f 1 | ↩
  xargs -I {} svmap {}
```

connects the results from Nmap directly to `svmap` to analyze open ports only.

`svwar` and `svcrack`

For even deeper analysis, you can combine `svmap` with other tools from the SIPVicious toolkit:

- `svmap`: This module can be used to identify SIP servers on the network. For example, you can scan an entire subnet with the `svmap 10.0.0.0/16` command to find all the accessible SIP servers. The output contains details of the IP addresses and ports used by the SIP servers.
- `svwar`: This module checks which extensions can be registered on a SIP server (e.g., `svwar 192.168.1.103`). The output shows which extensions are available and whether authentication is required.
- `svcrack`: This module can be used to crack SIP passwords by brute force or a dictionary file. For example, test the password strength for extension 100 to identify weak passwords and replace them as needed with the command

```
svcrack 192.168.1.103 -u 100 ↩
  -d passwords.txt
```

These tools can also be a big help if you suspect that unauthorized devices are accessing your SIP server. After a scan, you can use `svwar` to identify all registrable extensions, then `svcrack` to check whether passwords are strong enough. If you do find security vulnerabilities, you can change the passwords and enable mechanisms such as IP whitelisting or authentication certificates.

Conclusions

SIPVicious, `sngrep`, `sipsak`, Wireshark, and other open source tools can help you discover vulnerabilities in VoIP networks; fix typical problems, such as registration failures or interrupted connections; and generally boost the security and functionality of the infrastructure.

Info

[1] sngrep: [https://github.com/irontec/sngrep]
[2] Wireshark: [https://www.wireshark.org]
[3] sipsak: [https://github.com/nils-ohlmeier/sipsak]
[4] SIPVicious: [https://github.com/EnableSecurity/sipvicious]

The Author

Thomas Joos is a freelance IT consultant and has been working in IT for more than 20 years. In addition, he writes hands-on books and papers on Windows and other Microsoft topics. Online you can meet him on [http://thomasjoos.spaces.live.com].

Collaboration with Microsoft Loop

In the Loop

Microsoft Loop is a flexible collaboration tool that aims to redefine seamless integration and dynamic teamwork. We highlight the tool's features and components and show how real-time collaboration can be improved. By Philip Lorenz

The world of work is characterized by increasing digitalization and remote working, which presupposes flexible and dynamic collaboration tools. Many organizations are already using proven Microsoft tools such as Teams, OneNote, or Planner, but they reach their limits when it comes to spontaneous, temporary workflows or the seamless integration of content. This domain is precisely where Microsoft Loop enters the scene.

Key Idea Behind Loop

Microsoft Loop was first introduced at the end of 2021 and complements the Microsoft 365 universe as a platform for dynamic collaboration. Unlike traditional tools, Loop enables a new type of integration and real-time collaboration. Content such as notes, tables, or task lists can be flexibly created, directly embedded into other applications, and collaboratively edited – without media breaks.

Loop components are a key element in Loop. They can be added to chats in Teams or email in Outlook to support synchronization and real-time collaboration, regardless of the app in which they are used.

Loop is designed to offer teams a flexible and uncomplicated approach to working on shared tasks and is ideal for spontaneous coordination, brainstorming, or short-term project planning. Microsoft 365 users in particular benefit from seamless integration with applications such as Teams, Outlook, and Planner.

Microsoft Tool Comparisons

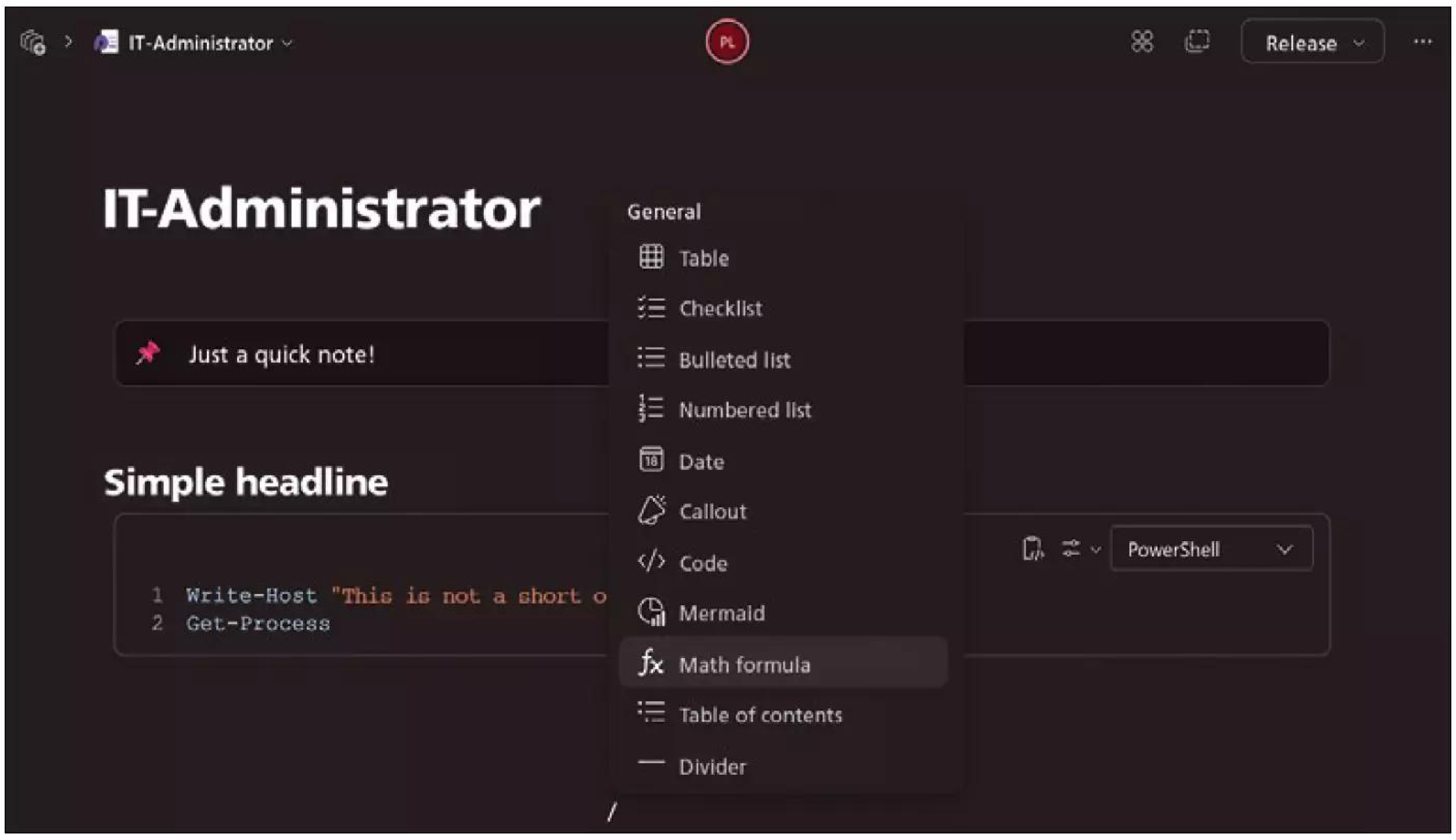
For many who are new to Microsoft Loop, one question naturally comes to mind. When does it make sense to use Loop? The boundaries between the various collaboration tools in the Microsoft 365 universe often seem fluid, and it is not always easy to identify which tool is best suited to which situation. However, this kind of classification is essential, especially for Loop newcomers, to reap the full benefits of the tool. Loop occupies a very specific niche and complements existing offerings:

- OneNote is used for the long-term organization of notes and information, offers structured content and excellent archiving, but is less suitable for dynamic, fast-paced collaboration.
- Planner offers clear responsibilities and deadline tracking with features such as Kanban boards, and is ideal for projects that require long-term planning and transparent progress monitoring.
- Loop focuses on flexible, temporary collaboration. The tool is ideal for spontaneous workflows and agile coordination that do not require a long-term structure (e.g., brainstorming, ad hoc project coordination, or collaborative work on documents).

Smart Features

An open Loop page is the central workspace for creative and dynamic collaboration. Getting started is very easy: You can start typing straightaway, similar to a classic text editor.

Lead Image © fabio formaggio, 123RF.com



IT-Administrator

Just a quick note!

Simple headline

```
Write-Host "This is not a short o
Get-Process
```

Figure 1: Typing "/" in the text opens a list showing components that can be embedded.

Table 1: Formatting

Element	Keystroke	Function
Headings	#	Followed by a space, a single hash symbol creates an H1 heading
	## and ###	Followed by a space, create sub-headings
Highlights	Ctrl+B (Cmd+B)	Bold
	Ctrl+I (Cmd+I)	Italic
Dividers	–	Creates horizontal lines
Components	/	Opens the menu to add Loop components
Mentions	@	Includes team members

Loop offers far more than simple text entry, though: Whether you need meeting notes, project ideas, or to-do lists, everything can be found directly on the Loop page (Figure 1). Thanks to support for markdown-style commands, it is easy to structure the text (Table 1).

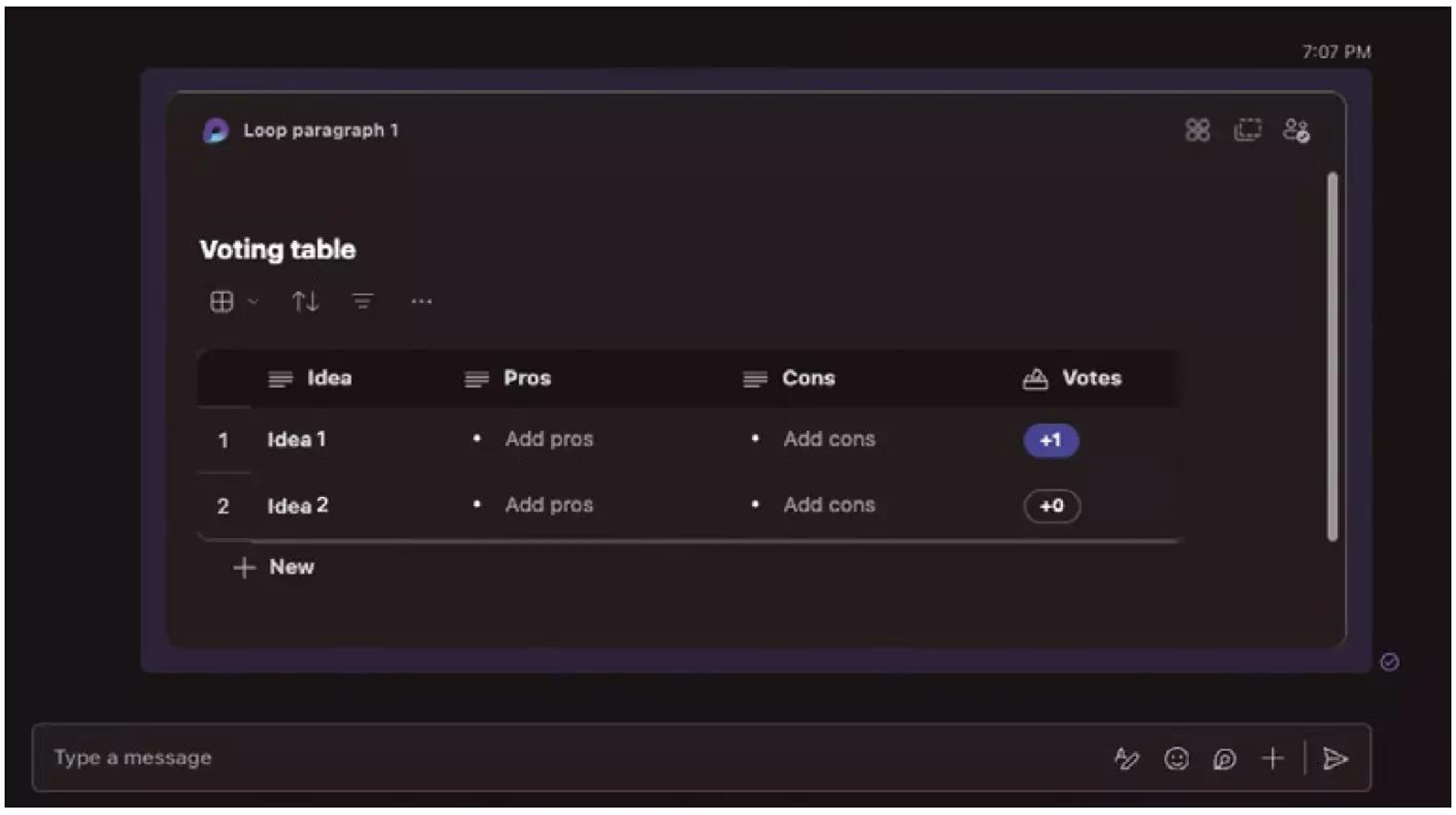
When adding components, you can insert lists, tables, and tasks – or even Planner items – and edit them seamlessly with your team. Mentioning team members is an ideal way to assign tasks or ask for input on specific topics. People addressed in this way receive a notification and can immediately take action on the page.

Components, Pages, and Workspaces

Components, pages, and workspaces form the basis for teamwork in Loop. Components are reusable building blocks, such as lists, tables, or notes – they stay synchronized regardless of whether you are using them in Teams, Outlook, or Whiteboard. Components really come into their own for short-term tasks or brainstorming sessions. Loop components are particularly useful in Teams; they can easily be added to a chat – for example, to conduct a vote (Figure 2).

As the name suggests, Loop pages are individual pages that contain text and, where needed, several components. Their structure and classification within the overall architecture are comparable to OneNote pages. Microsoft provides a collection of templates to help users create pages. From project overviews and one-to-one meetings through competitive analysis, you will definitely find some useful stuff on board.

Finally, workspaces provide a central container for organizing projects and bringing all relevant information together in one place. Teams can collect and structure documents, tasks, and communication content in a workspace. Breaking things down into pages empowers users



Voting table

	Idea	Pros	Cons	Votes
1	Idea 1	Add pros	Add cons	+1
2	Idea 2	Add pros	Add cons	+0

Figure 2: Loop components include polls and votes, which can also be integrated into Microsoft Teams.

to present project plans, progress reports, and brainstorming results in a clear-cut way. All of the project information is available centrally and easily accessible, without having to switch between different tools. Files, tasks, and resources are readily available within the workspace, and edits and additions are immediately synchronized.

In addition to Loop's internal components, elements from other programs can also be easily imported and displayed, removing the need for users to switch constantly between different tools – primarily Microsoft products, of course. For example, a Planner board can easily be inserted and is then handled like any other component in Loop. In addition to Microsoft's offerings, other external systems can be integrated, such as Jira, Trello, or Figma.

Planning and Documenting Meetings

With its flexible features and templates, Microsoft Loop offers useful support for planning and conducting meetings, and recording the results in a sustainable way. The ability to create meeting notes as Loop pages directly in tools such as Teams or Outlook is amazingly practical.

The use of ready-made templates makes it far easier to structure and organize meetings. In just a few steps, you can create a Loop page that groups the agenda and the meeting notes.

Before you hold a meeting, you can create a Loop page for the planned meeting in Teams or Outlook and add the agenda to give participants a clear overview of the topics. Next, share the Loop page with the invitees so they can add comments or make other useful contributions up front.

During the meeting, participants can write and add to the page in parallel, creating a common understanding. Tasks that arise from the discussions can be integrated directly into the Loop page and assigned to the appropriate people.

After the meeting, the Loop page is kept as documentation and can be accessed by all the participants at any time. Participants can add open items or updates without losing any content. Last but not least, the notes are searchable with Copilot.

Effective Loop Management

Loop offers powerful collaboration tools but also presents new challenges in terms of governance, storage management, and access authorizations. To begin, you need to activate Loop in the Microsoft 365 Admin Center (*admin.microsoft.com*) under *Settings* | *Org settings* | *Services*.

Workspaces and components can be activated separately. A cloud policy, which can be created in the Microsoft 365 Apps admin center (*https://config.office.com/*), lets you define which user groups have access to which features.

When it comes to the storage type and locations of Loop elements, you can assume the following basic principles: Loop components and pages are stored as files with the `.loop` extension and are similar in terms of handling to other Office file formats such as DOCX, PPTX, or XLSX. These files benefit from features such as versioning, audit logs, change tracking with author information, export options, eDiscovery workflows, and legal blocking.

Loop workspaces are much like containers and are comparable to SharePoint pages. You can apply governance and compliance policies to workspaces and use tools such as AvePoint or ShareGate for management. A shared Loop workspace gives authorized users access to all pages and linked files, provided they have the appropriate access authorizations for the underlying files. With the SharePoint Embedded Administrator role, IT admins can access and manage workspaces in containers in SharePoint Admin Center. Microsoft announced a container search feature for the first quarter of 2025.

Storage Locations

The choice of storage locations for Loop data is not entirely consistent and can present a challenge in terms of administration. In many cases, it is up to the user to decide where data is stored – often without the user being aware of the fact.

The storage location is selected when a component is created; the advantage of being able to create loop components across applications can quickly become a problem. **Table 2** provides information about the storage location according to the initial creation location.

Problems arise when an employee leaves the organization but still has shared components stored in their OneDrive. Many users are aware that shared Office documents must be transferred from OneDrive, although it's probably not so obvious in the case of Loop components.

Personal content such as meeting notes or Loop components from Teams chats are stored in the creator's OneDrive. When an employee leaves the company, OneDrive's standard deletion policies apply: Content remains active for 30 days, is then soft deleted for another 93 days, and is finally removed. During this time, you can access and back up the content in the SharePoint Admin Center.

Table 2: Loop Storage Locations

Initial Creation Location	SharePoint Embedded	SharePoint Page	User's OneDrive
Loop app	x	–	–
Copilot pages	x	–	–
Teams channel meeting	–	x	–
Teams channel	–	x	–
Teams (private chat)	–	–	x
Teams (private meeting)	–	–	x
Outlook email	–	–	x
OneNote	–	–	x
WhiteBoard	–	–	x

The situation is different for shared workspaces stored in SharePoint Embedded. They remain in the Microsoft 365 tenant after an employee leaves but are then considered "ownerless," which means that no new owner can be assigned and the workspace is kept but not actively managed. This situation can lead to unused resources, which is why you need to perform regular checks of these workspaces with PowerShell commands such as `Get-SPOSite`.

Copilot pages created directly by Microsoft Copilot are a special case. This content is linked to the user account and is automatically deleted as soon as the associated user account disappears. Transfer or persistent storage is not possible, and you need to create manual backups at an early stage to archive any content the organization wants to keep.

The recommendation here is to establish clear-cut processes for handling Loop content when employees leave the company, including transferring content before the employee disappears and using tools to identify and back up potentially critical data. A new feature in 2025 allows you to designate temporary administrators for user accounts during the retention period to back up or transfer important content.

Inside and Outside

Sharing Loop content within the organization is a seamless experience that supports collaboration in Teams environments. Employees can share Loop components and workspaces directly in Teams chats, channel discussions, or Outlook email. All changes are synchronized in real time, regardless of where the content is accessed.

Loop also lets users share workspaces and components with external users, including guests. However, a few important requirements include: (1) registered guests, (2) sensitivity labels, and (3) specific sharing links. External users must be registered as B2B guests in the tenant database to access Loop content. If sensitivity labels have been applied to workspaces or components, sharing with external users is blocked to protect sensitive data. Sharing is only possible with defined individuals through *Share with specific people* links. General sharing links are not available to external users.

Conclusions

Microsoft Loop brings a breath of fresh air to the world of collaboration tools, especially for businesses. The tool impresses with seamless integration, real-time collaboration, and flexible usage options, whether in Teams, Outlook, or other Microsoft applications. The possibilities for efficiently implementing spontaneous workflows and temporary projects are particularly exciting.

Some challenges still remain: Administrators need to keep track of storage locations, access rights, and governance, and users need clear-cut classification within the existing tool mix. However, given smart configuration and targeted training, Loop can unleash its full potential and play a decisive role in today's world of work. ■

Simulate network traffic with open source tools

Plan of Action

Network simulators provide the information admins need to plan and grow wired and wireless networks.

By Holger Reibold

Digitilization of the economy is leading to a rapidly increasing demand for powerful IT infrastructures in almost all industries. While some seek salvation in the cloud, others are assessing their needs and trying to ensure that their own infrastructure is sufficiently scalable to cope with sudden spikes in demand.

Although basic network standards have hardly changed in recent decades, the increasing volume of traffic is posing significant challenges for organizations. To ensure reliability, it's important for infrastructures to have a clear-cut design that offers scope for expansion. Network simulation helps admins with the demanding tasks this responsibility involves. In practice, this technology can be used in various contexts:

- Replacing physical networks: Physical networks are difficult to instantiate, although it makes sense to experiment with scenarios that are as concrete as possible. Visual tools help create subnets or remote networks with just a few clicks.
- Early prototyping: Industrial applications in particular require the ongoing development of existing protocol families. Simulators allow protocols, such as those promoted by ODVA (formerly Open DeviceNet Vendors Association, Inc.), to be tested and allow users to familiarize themselves with the protocols. In this way, organizations can gain initial experience with the Common Industrial Protocol (CIP), the Open Screen Network Protocol, or the Open Screen Application Protocol, for example, and modeling facilitates planning and testing.
- Error analysis and optimization: Simulations help identify potential bottlenecks and issues before they materialize. They also provide clues for the best possible load distribution configuration and resource utilization. Additionally, insights for optimizing security can be derived.

Creating virtual duplicates of existing networks to recreate specific scenarios and develop a better understanding of specific phenomena also makes sense. Creating duplicates of existing networks is interesting from a security perspective because it lets you simulate or reconstruct attacks and test the sensitivity of the network topology.

Finding the Right Tool

Over the past decades, a variety of tools for traffic simulation have emerged. They differ in terms of protocol support, implementation approaches, degree of extensibility or scalability, and execution speed. Additionally, a distinction must be made between simulators and emulators. Another distinguishing feature is that not every simulator can replicate wireless networks.

Basically, the choice of simulator depends on your requirements, and it makes sense to focus on specific application scenarios. In practice, latency times or the scalability of multicast protocols will tend to be of interest. Where video conferencing tools are seeing increasing use, it's important to approximate the jitter in video applications.

Most network simulation environments use discrete event simulation methods, which means that only the state of the simulated network at discrete points in time is of interest, and the states between these nodes are irrelevant. In reality, the processes are far more complex: When a data packet is sent, each individual bit must be encoded by electrical or optical waveforms, followed by computations for error detection and error correction information to verify the

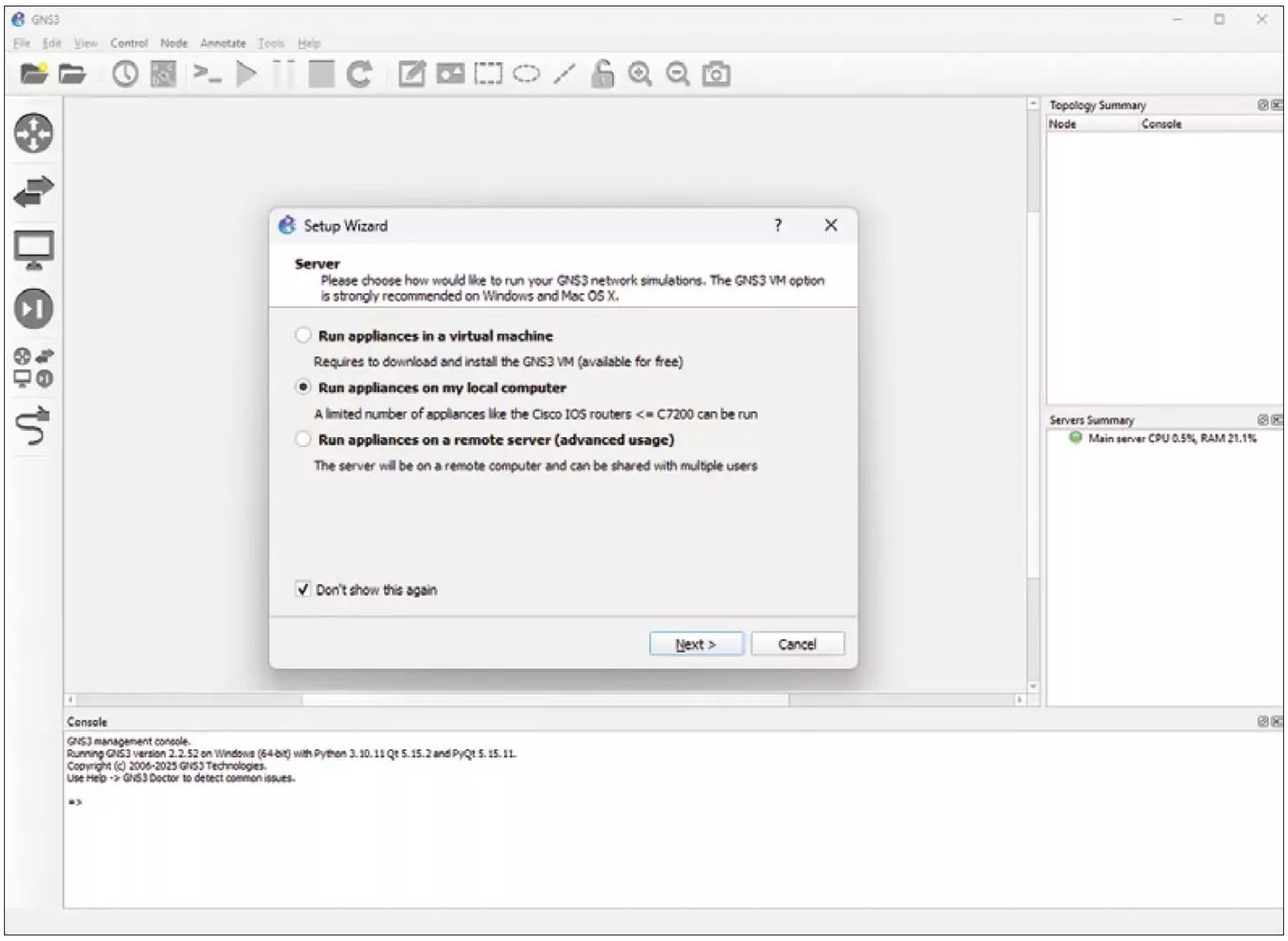
Photo by Jakub Berdzicki on Unsplash

correctness of the received signal, reconstruct it at the receiving end, and far more.

Network simulators benefit from these simplifications because they only replicate the relevant nodes and their properties and functions. A computer network usually comprises a varying number of nodes; typical components include consumer systems, routers, Ethernet and WiFi connections, protocols, data packets, and various applications. To replicate these structures, simulators generate specific objects and assign the desired properties to them. In addition to node and interface objects, link objects ensure the connectivity of the various components.

The choice of a suitable simulator is influenced by various factors. Besides the general capabilities, such as object, protocol, and simulation options, aspects such as a GUI, statistical functions, and evaluation play important roles. Various tools, especially emulators, also allow real and simulated networks to be linked.

The simulation of wireless networks poses a challenge because the models are difficult to generate and validate (e.g., replicating the broadcast properties of wireless networks). The learning curve that awaits you when using a traffic simulator should also be taken into account when you make your selection.


GNS3
File Edit View Control Node Annotate Tools Help
Topology Summary
Node
Console
Servers Summary
Main server CPU 0.5%, RAM 21.1%

Setup Wizard

Server

Please choose how would like to run your GNS3 network simulations. The GNS3 VM option is strongly recommended on Windows and Mac OS X.

○ **Run appliances in a virtual machine**
Requires to download and install the GNS3 VM (available for free)

● **Run appliances on my local computer**
A limited number of appliances like the Cisco IOS routers <= C7200 can be run

○ **Run appliances on a remote server (advanced usage)**
The server will be on a remote computer and can be shared with multiple users

☑ Don't show this again

Next > Cancel

Console

```
GNS3 management console.
Running GNS3 version 2.2.52 on Windows (64-bit) with Python 3.10.11 Qt 5.15.2 and PyQt 5.15.11.
Copyright (c) 2006-2025 GNS3 Technologies.
Use Help -> GNS3 Doctor to detect common issues.

=>
```

Figure 1: Configuring GNS3 as a local environment means that you can run it on your own computer.

The Ancestor, ns-2

One of the first simulators to gain widespread acceptance was the open source ns-2 [1], which was developed and is maintained by the Information Sciences Institute (ISI) at the University of Southern California. This simulator emerged from the REAL simulator [2] in 1989 and was further developed with the support of DARPA. Operations primarily take place at the command line, but various GUIs, such as Visual Network Simulator [3] or nsBench [4], can be used. The ns discrete event simulator uses its own scripting language, TCL, to specify simulation topologies and applications. Version 3 of this simulator is now available. GTNetS (Georgia Tech Network Simulator) [5] takes things one step further, with its design reflecting the network protocol stack and object properties.

In the discussion about which simulator is recommended, the same advice continually pops up: Some of these tools are open source, some are free. Cisco Packet Tracer [6], Cisco CML [7], EVE-NG [8], and PNETLab [9] are highly recommended. However, the license under which each is available isn't always clear. For example, GNS3 is subject to GPL v3.0, while the licensing of PNETLab is unclear. OMNeT++ [10] is mentioned far less frequently, but this event simulator is only free for academic use; special licenses are required for commercial use.

The scientific community provides guidance on the selection process. A large number of experts advocate the use of ns-2 and ns-3; however, both ns variants are considered challenging to use. GNS3 [11] is one of the open source tools that offers great functionality and flexibility, as well as an active community (Figure 1).

Getting Started with GNS3

GNS3 is one of the most widely used specialist simulation and emulation tools. GNS (graphical network simulator) has been in development since 2006. You can use it to develop smaller network topologies, as well as implement extensive server farms. For particularly tough requirements, you can also host GNS in the cloud. Jeremy Grossmann, the original GNS3 developer, programmed the software to prepare for his Cisco Certified Network Professional (CCNP) certification.

GNS3 now supports every common type of network device. Software-defined networks (SDNs), network functions virtualization (NFV), any type of Linux server, and Docker installations can be simulated. The environment is based on three software components: the GNS3 all-in-one software, the GUI, and the virtual machine. You create the desired topologies in the GUI, and the generated devices are hosted by a server process that's available as a local GNS3 server or as a virtual machine (VM).

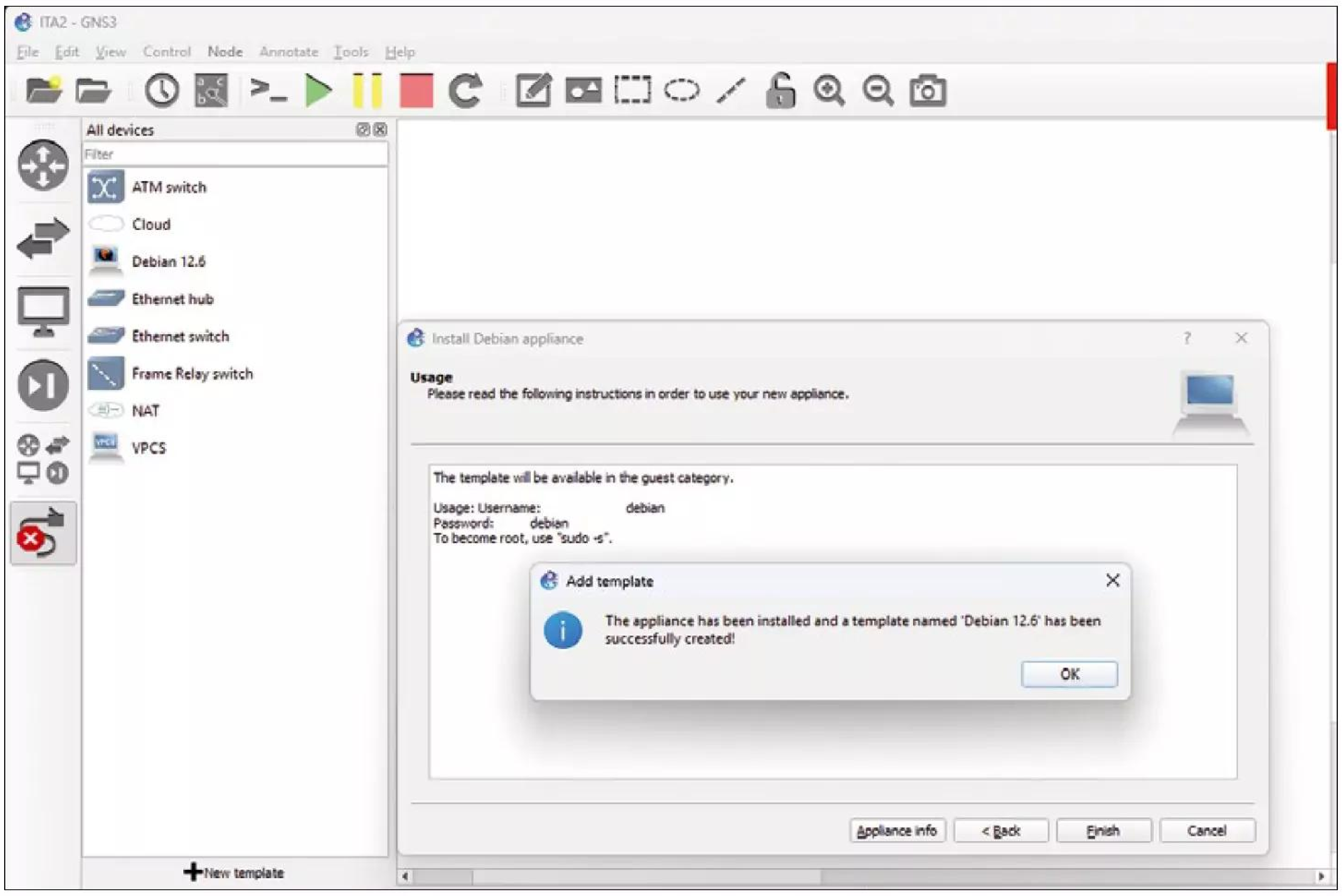
Before creating your first network components, you need to be aware of another special GNS3 feature: The environment supports, emulates, and simulates devices. As an emulation, GNS3 mimics the device hardware; to do this, you run an image on the virtual device. Simulation takes a different approach, with GNS3 replicating the functions of a device. In practice, the two techniques often overlap. A large number of predefined components are available for downloading from the project website or GitHub. In principle, you can use as many components as you like. The only limitations are the available computing capacity and the execution variant.

Also of benefit is that GNS3 enables real-time network simulation without network hardware to support testing before deployment. The simulator also supports dynamic network cards for performing proof-of-concept testing. You are by no means limited to simulation or emulation; you can connect GNS3 to a physical network to expand your lab environment.

Another benefit is that the client and server are available for Linux, macOS, and Windows. To set up a GNS3 environment on Windows, simply run the setup program, which also helps you install the VM server component. Various tools such as Wireshark and WinPcap are installed by default, and a web client can be installed, as well.

After installing GNS3, you need to connect to the GNS3 server, which you can run on a VM to keep things simple. The GNS3 client gives you a setup wizard for the configuration. After selecting the VM, the client opens the connection to the GNS3 server. Before starting the simulation, you might need to make specific devices available in GNS3. The client offers various standard devices from the *All devices* selection, but it won't get you very far in practice. To make additional devices available in the GNS3 client, click *New templates*. The local GNS3 server offers various images. Alternatively, you can import existing appliances or generate your own templates.

In addition to the standard components, the GNS3 Marketplace offers appliances for Docker installations, Windows and Linux servers, and a wide range of Cisco devices. Of course, modules for typical client operating systems and mobile devices are available (Figure 2), as well.



Usage

Please read the following instructions in order to use your new appliance.

The template will be available in the guest category.

Usage: Username: debian
Password: debian
To become root, use "sudo -s".

The appliance has been installed and a template named 'Debian 12.6' has been successfully created!

Figure 2: A new network component is created in GNS3 by template management.

Creating a Simple Network Topology

Once you have expanded object management to include the required network components, it's time to create an initial network topology. The procedure is simple. The first step is to open a new project from the File menu.

On the layout screen, start by dragging the desired object types into the workspace. The various network components are available on the *Device* toolbar, which differentiates between routers, switches, end devices, and security components. You will find both simulated and emulated devices in the various groups. The Topology Summary area in the upper-right corner of the GNS3 client lists all the nodes of the current topology. The summary also shows which components have been launched and which have not.

The topology starts to take shape as you add each new component.

Besides typical network components such as routers, switches, and so on, PC instances are particularly in demand. GNS3 provides the *VPCS* (virtual PC simulator) type for this purpose. The PCs you add are labeled in sequence as PC1, PC2, and so on.

You can edit the entries in the configuration menu. To do so, right-click on the desired object and run the *Configure* command from the context menu. You can set up connections between the various network objects with the *Add a link* function. To connect a PC object to a switch, first enable the link function, then click on the corresponding VPCS entry; next, select the *Ethernet_x* connection type and click on the switch component. New PCs are not active by default, but you can change this setting by right-clicking to pop up the context menu and running the *Start* command.

The next step is to make further adjustments to the environments and, for example, to assign specific IP addresses to the PCs as needed. To configure the various devices in sequence, click the *Console connect to all devices* button in the toolbar. Depending on the GNS3 client operating system used, the simulator opens a terminal emulator. A PuTTY variant is installed by default on Windows (**Figure 3**).

Use the `ip <IP address> <subnet>` command to assign the desired address to the device. GNS3 does not offer any way of simplifying this process. Then run `ping` to check whether the PCs can communicate with each other. To save the configuration, run `save` at the command line. Nodes to which you have assigned a static IP address will show this address in the topology view.

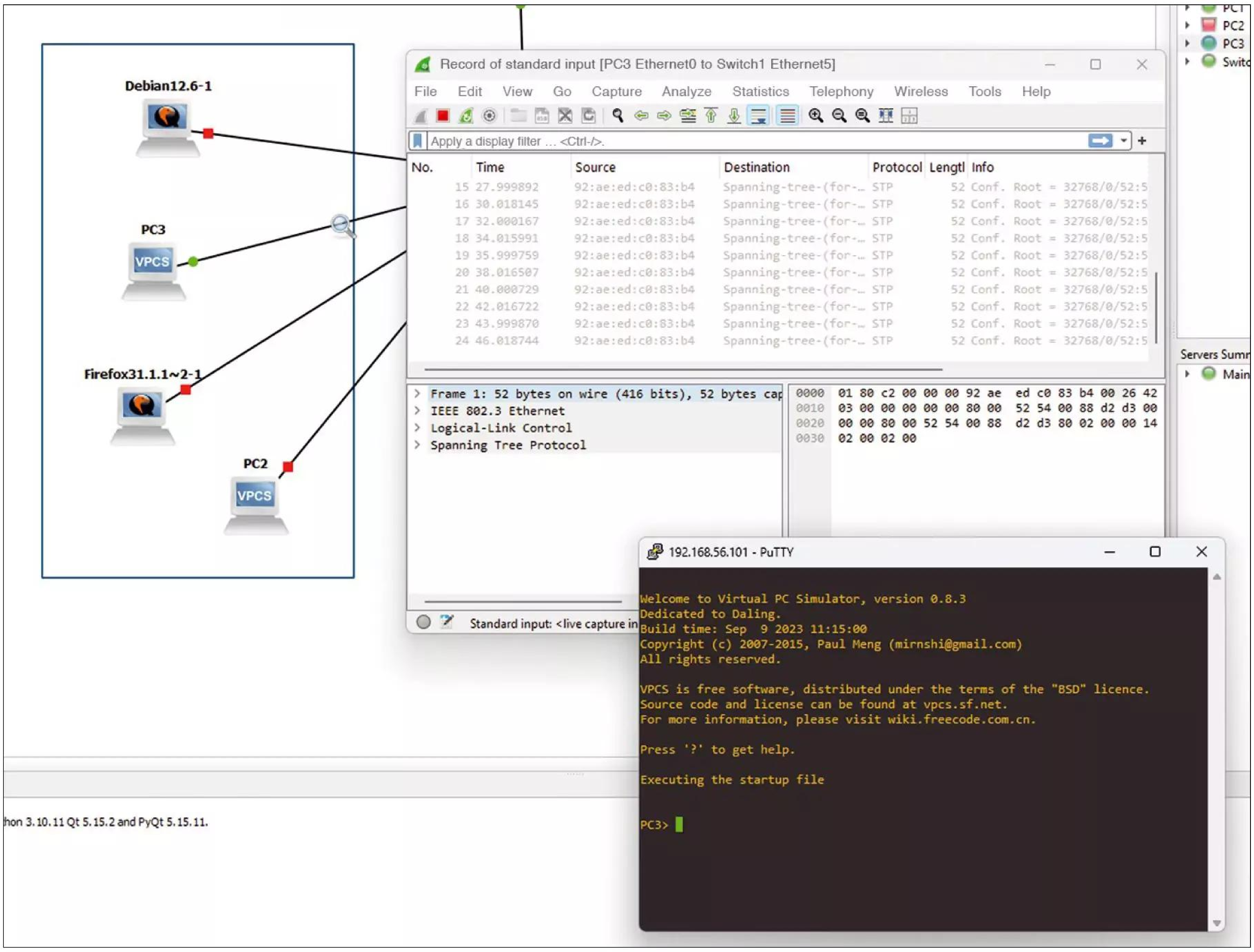
For a long time, GNS3 did not offer WiFi support, but this has now changed. To simulate WiFi, you can create a *Cisco Virtual Wireless Controller*, for example. In principle, though, any other common controller can also be used; to do this, you simply need to create a new template and add a suitable image, although finding the right images can be difficult.

Advanced Techniques

Thanks to its long tradition, GNS3 offers impressive flexibility and functionality. It is particularly easy to replicate networks and network segments. Admins benefit in particular from the great scalability; only the available resources limit the number of network nodes and therefore the size of the simulated network. The fact that you can precisely replicate real network conditions is hugely useful.

GNS3 supports various customization options for network objects. After right-clicking on an object, you can start, stop, or reload it. Additionally, various symbols can be assigned to the objects. Creating layers and duplicating node configurations is also easy, and the ability to integrate third-party applications such as Wireshark is another selling point. The Wireshark sniffer lets you record the traffic in your simulated environment. To do so, select a connection and click on the magnifying glass icon – Wireshark will immediately start logging.

One advantage of using GNS3 is that you can extend the simulated network to include an Internet connection. For a local GNS3 installation, first add a cloud to the GNS3 topology. Then adjust the IP addressing, DNS triggering, and NAT configuration. The developers introduced network address translation (NAT) mode in version GNS3 v2.0. The benefit of the Internet connection is obvious: It means you can put the simulated environment through its paces under


Debian12.6-1
PC3
VPCS
Firefox31.1.1~2-1
PC2
VPCS
PC1
PC2
PC3
Switc
Servers Sumr
Main
Record of standard input [PC3 Ethernet0 to Switch1 Ethernet5]
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Apply a display filter … <Ctrl-/>.
Standard input: <live capture in
192.168.56.101 - PuTTY
hon 3.10.11 Qt 5.15.2 and PyQt 5.15.11.

```
No.     Time        Source              Destination            Protocol Lengtl Info
     15 27.999892   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     16 30.018145   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     17 32.000167   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     18 34.015991   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     19 35.999759   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     20 38.016507   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     21 40.000729   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     22 42.016722   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     23 43.999870   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
     24 46.018744   92:ae:ed:c0:83:b4   Spanning-tree-(for-…  STP          52 Conf. Root = 32768/0/52:5
```

```
> Frame 1: 52 bytes on wire (416 bits), 52 bytes cap
> IEEE 802.3 Ethernet
> Logical-Link Control
> Spanning Tree Protocol
```

```
0000  01 80 c2 00 00 00 92 ae  ed c0 83 b4 00 26 42
0010  03 00 00 00 00 00 80 00  52 54 00 88 d2 d3 00
0020  00 00 80 00 52 54 00 88  d2 d3 80 02 00 00 14
0030  02 00 02 00
```

```
Welcome to Virtual PC Simulator, version 0.8.3
Dedicated to Daling.
Build time: Sep  9 2023 11:15:00
Copyright (c) 2007-2015, Paul Meng (mirnshi@gmail.com)
All rights reserved.

VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file


PC3>
```

Figure 3: **The additional tools integrated into GNS3 are indispensable for configuring the simulated devices and for traffic analysis.**

real-world conditions. Basically, various GNS3 servers can be controlled in the GNS3 GUI. The profile function is available for this purpose; you can access it under *Edit | Preferences*.

Expanding the Toolbox

GNS3 does a great job for standard tasks, but the environment can seem a little antiquated at times. If you want a quick and easy solution, PNETLab (a packet network emulator tool) is a good option. This platform lets you create network simulations and share these configurations with the community (Figure 4). The PNETLab developers refer to simulated configurations as "labs."

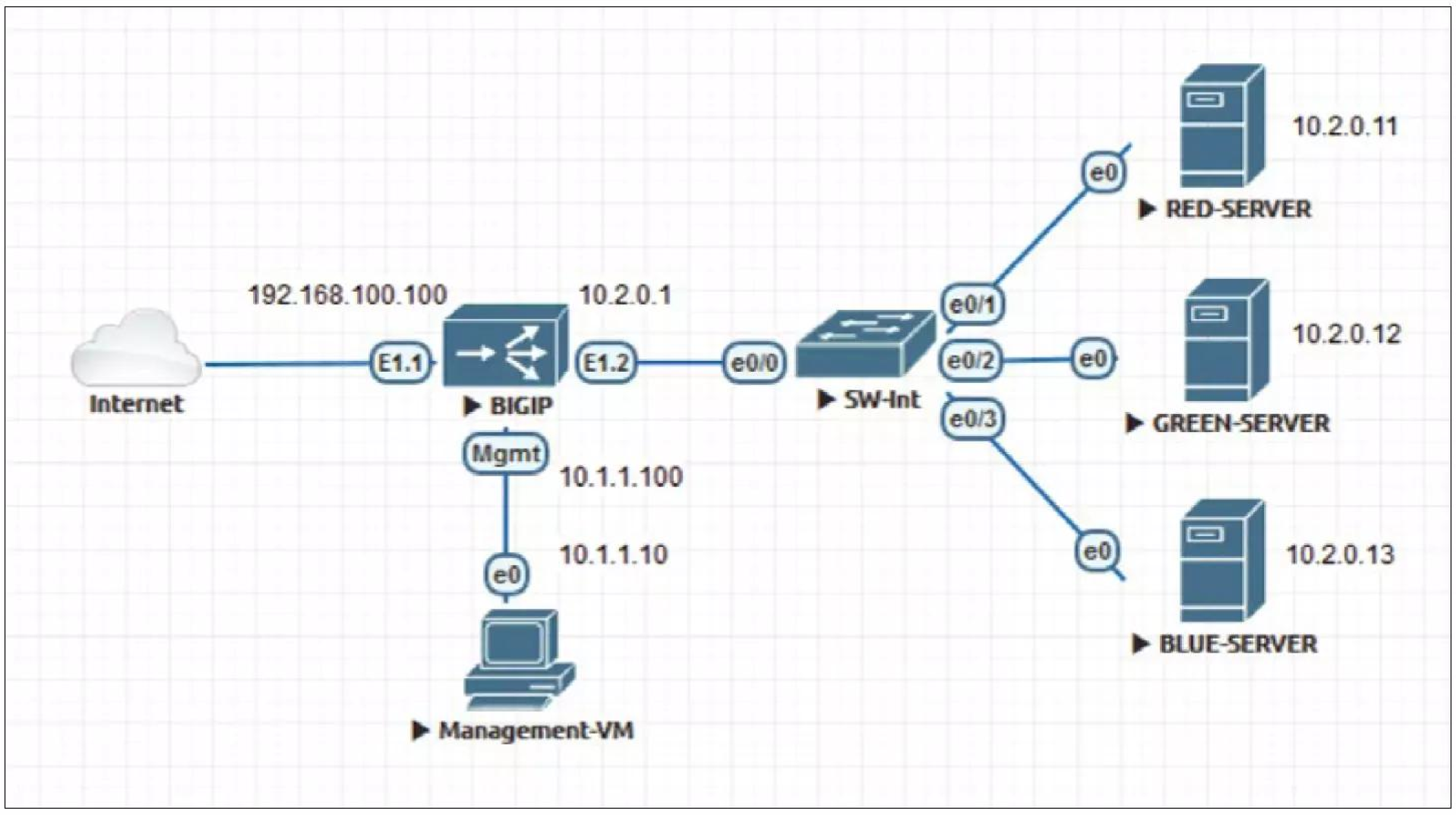


Figure 4: A network configuration created with PNETLab.

The PNETLab Box is the heart of the environment – a virtual machine in which the simulation is created and executed. PNETLab supports both online and offline modes. Online mode requires an existing Internet connection, but also means that you can take advantage of existing lab configurations.

The procedure for network simulation is essentially the same as for GNS3 and comparable tools: You create a lab and populate it with the desired network components. The next step is to create the connections. The *Add a new node* command gives you dozens of components from which to choose. You can also configure the node in the matching dialog box.

The Future Role of AI

Given the media hype surrounding AI, you might assume that application scenarios in information technology in particular would benefit from technological progress. However, in areas where artificial intelligence should feel at home (e.g., in IT and network administration), it has played a fairly restricted role thus far. Plainly, network simulators will have AI functions in the future, and the first commercial products are in the pipeline, including Ekahau AI Pro [12], for the design of wireless networks.

These systems promise automated problem detection, with much to suggest that the new features will boost efficiency and offer more accurate forecasting. Troubleshooting is also likely to become easier. On the basis of current knowledge, it's difficult to say when AI support can be expected in open source simulators. One fundamental problem with the AI discussion is the lack of empirical evidence. To date, neither scientifically sound impact assessments nor empirical evidence exists regarding the effectiveness of AI, which might be the missing piece of the puzzle.

Conclusions

Modern IT infrastructures need to meet increasingly complex requirements and be geared to handle greater loads in the future, which means that planning and benchmarking are becoming increasingly critical tasks. Tools that specialize in network traffic simulation offer valuable support in this regard by helping admins recreate and test structures. A wide range of options is available, some of the most interesting tools being GNS3 and PNETLab. As always, though, it is important to check which specialist tool best meets the specific requirements of your own use case. ■

Info

[1] ns-2: [https://www.isi.edu/websites/nsnam/ns/]
[2] REAL: [https://www.cs.cornell.edu/skeshav/real/overview.html]
[3] Visual Network Simulator: [https://apus.uma.pt/vns/]
[4] nsBench: [https://www.andrew.cmu.edu/user/eshaer/courses/resources/NSBenchDocs/index.htm]
[5] GTNetS: [https://ieeexplore.ieee.org/document/1261483]
[6] Cisco Packet Tracer: [https://www.netacad.com/cisco-packet-tracer]
[7] Cisco Modeling Labs: [https://www.cisco.com/site/us/en/learn/training-certifications/training/modeling-labs/index.html#tabs-a107e9a621-item-b27a5c1e52-tab]
[8] EVE-NG: [https://www.eve-ng.net]
[9] PNETLab: [https://www.pnetlab.com/pages/main]
[10] OMNeT++: [https://omnetpp.org]
[11] GNS3: [https://www.gns3.com]
[12] Ekahau AI Pro: [https://www.ekahau.com/rsc/network-simulator-ekahau-ai-pro/]

The Author

Holger Reibold, computer scientist, has worked as an IT journalist since 1995. His main interests are open source tools and security topics.

SCALE 23X

Where Open Source Meets

Join North America's largest community-run open source conference at SCALE 23x in Pasadena, CA — March 5-8, 2026.

Four days of sessions on Linux, DevOps, cloud native, and security.

Connect with thousands of developers and open source enthusiasts celebrating free and open source software.

Register Now!

register.socallinuxexpo.org/reg23

COLOCATED EVENTS:
AstriCon, Cloud Native Days LA, Open Source in Higher Education, PlanetNix, DevOpsDay LA, SunSecCon, Open Source Career Day, SCaLE Youth

Get 30% off with Promo code: AD30

socallinuxexpo.org

SCALE 23X
MAR 5-8, 2026

An AI developer workflow with Ollama+Cursor

Privacy First

We explore an on-premises, open source AI developer workflow that uses Ollama+Cursor for maximum data sovereignty and control.

By Vidhu Arora, Shreyan Gupta, and Dr. B. Thangaraju

The integration of large language models (LLMs) into the software development lifecycle has become essential for boosting developer productivity. This shift presents a critical choice for technical leaders: achieving absolute data sovereignty with an on-premises, open source solution. In this article, we provide a comprehensive analysis of this philosophy, offering a deep technical guide to making a strategic decision.

The on-premises, privacy-first stack combines Ollama [1] for local LLM serving with the Cursor [2] artificial intelligence (AI)-native integrated development environment (IDE). This approach is architected for maximum control, ensuring that proprietary code, developer prompts, and model outputs never leave the organization's network perimeter. The system represents the pinnacle of data privacy, trading the operational overhead of managing physical infrastructure for unparalleled security and a significantly lower total cost of ownership (TCO).

This guide delves into the architecture, implementation, performance benchmarks, cost projections, and security considerations of the Ollama stack by providing actionable implementation details, including hardware build lists, Docker configurations, and security hardening scripts. Through a data-driven analysis, we illuminate the benefits of control, low latency, and capital expenditure. The analysis culminates in a strategic framework to help organizations determine whether this workflow aligns with their priorities regarding privacy, performance, budget, and technical expertise.

Ollama+Cursor

The on-premises workflow represents the definitive choice for organizations where data privacy and control are non-negotiable. By containing the entire AI-assisted development loop within the organization's own infrastructure, it eliminates the risks associated with third-party data handling and external network exposure. This section provides a deep dive into the architecture, implementation, and security of the Ollama + Cursor stack.

The elegance of the on-premises stack lies in its architectural simplicity and the clear, auditable data flow it provides. Every component is designed to operate locally, ensuring that sensitive intellectual property remains under the organization's exclusive control. The following component analysis describes how the architectural design provides a privacy guarantee that is not merely based on policy or contractual agreements but is a fundamental property of the system's operation.

Ollama Core Architecture

Ollama functions as a lightweight, self-contained LLM serving platform built on a classic client-server model. When initiated by the `ollama serve` command, it starts an HTTP server that listens for API requests – by default on *http://localhost:11434*. This server acts as a local endpoint, managing the loading, execution, and unloading of language models in response to API calls from clients like Cursor or custom applications.

The Inference Engine – llama.cpp

At its heart, Ollama is a sophisticated and user-friendly wrapper for the highly optimized `llama.cpp` inference engine. This C++-based engine is renowned for its performance and efficiency, particularly in running quantized models on a wide range of hardware, including consumer-grade GPUs and even CPUs. This foundation is what makes local LLM execution feasible, because `llama.cpp` handles the

Lead Image © bluebay, 123RF.com

complex mathematics of inference with remarkable speed.

Model Format – GGUF

The feasibility of running large, powerful models on local hardware is largely enabled by the generative pre-trained transformer (GPT)-generated unified format (GGUF). GGUF is a binary format specifically designed for storing and rapidly loading quantized language models. Quantization is a process that reduces the numerical precision of a model's weights (e.g., from 16-bit floating-point numbers to 4-bit integers), which dramatically shrinks the model's size and reduces its video random access memory (VRAM) footprint, allowing models that would otherwise require expensive data center GPUs to run effectively on consumer hardware, forming the cornerstone of the on-premises strategy.

Ollama Layered Model Storage

Drawing inspiration from containerization technology like Docker, Ollama employs a layered storage system for models. When a model is pulled, its components (e.g., weights, configuration, templates) are stored as individual "blobs" in the `~/.ollama/models/blobs/` directory. A "manifest" file, akin to a Docker image manifest, then defines how these blobs are assembled to form the complete model. This architecture is highly efficient, because different models or custom variants that share a common base layer can reuse the same blobs, saving significant disk space.

Developer Interface – Cursor IDE

Cursor is an AI-first code editor built as a fork of the open source Visual Studio Code (VS Code), ensuring a familiar developer experience while integrating deeply with LLMs. Its primary appeal for a privacy-first workflow stems from its robust data protection features.

Privacy-First Features

Cursor's *Privacy Mode* is a key feature that guarantees no code snippets or prompts are ever stored on its remote servers; it is technically enforced through mechanisms like the `x-ghost-mode` HTTP header, which routes requests from privacy-enabled users to a separate, log-restricted infrastructure. Furthermore, developers can disable codebase indexing, which prevents Cursor from creating and uploading summaries of the codebase for context, offering an additional layer of data containment.

Connecting to a Local LLM

The critical link that completes the private workflow is Cursor's ability to connect to a custom LLM endpoint. Within its settings, a user can override the default API endpoint (which points to services like OpenAI or Anthropic) and direct it to a local URL. By pointing to the local Ollama server's address, the entire AI interaction is contained within the local machine.

Data Flow

The data flow (Figure 1) within this architecture is entirely self-contained, providing an unparalleled level of privacy. From the moment a developer triggers an AI action to the moment code is generated, no data traverses the public Internet.

Cursor's *Privacy Mode* ensures code is not persisted on its servers, and the local endpoint configuration ensures the LLM request itself never leaves the local network. Ollama then processes this request with locally stored models. Consequently, the entire loop from developer keystroke to AI-generated code is air-gapped from external services, delivering a degree of data sovereignty and control that cloud-based solutions, by their very nature, cannot replicate, which is a crucial distinction for organizations handling highly sensitive intellectual property.

Deploying the on-premises stack is a straightforward process. The following sections provide a guide for setting up a robust and reproducible Ollama environment and integrating it with the Cursor IDE.

Deploying on a Linux Server

To set up the Ollama Server on Linux (Ubuntu 24.04), begin by installing Ollama. The recommended method is with the official installation script. This command downloads and executes the script, which handles the setup of the `ollama` binary and system service:

```
curl ↩
  -fsSL https://ollama.com/install.sh | sh
```

After installation, the script typically sets up Ollama to run as a systemd service, which ensures it starts automatically on boot. To start and enable the Ollama service and verify the server is running, enter

```
sudo systemctl enable --now ollama
sudo systemctl status ollama
```

Alternatively, for development or debugging, you can run the server directly in your terminal:

```
ollama serve
```

With the server running, use the `ollama pull` command to download

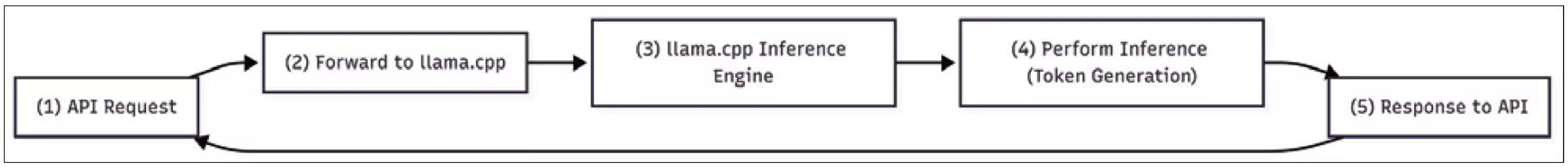

(1) API Request
(2) Forward to llama.cpp
(3) llama.cpp Inference Engine
(4) Perform Inference (Token Generation)
(5) Response to API

Figure 1: The data flow in the on-premises stack is entirely self-contained. The Cursor IDE sends a local API request to the Ollama server, which processes it with the `llama.cpp` engine and locally stored models, ensuring no data leaves the machine.

Listing 1: Docker Compose for Ollama

```
# docker-compose.yml
version: '3.8'
services:
  ollama:
    image: ollama/ollama:latest
    container_name: ollama
    ports:
      - "11434:11434"
    volumes:
      - ollama_data:/root/.ollama
    # To enable NVIDIA GPU access, ensure the NVIDIA
      Container Toolkit is installed
    # on the host machine and uncomment the following
      'deploy' section.
#  deploy:
#    resources:
#      reservations:
#        devices:
#          - driver: nvidia
#            count: all
#            capabilities: [gpu]
  restart: unless-stopped
volumes:
  ollama_data: {}
```

a model from the Ollama library. For a powerful coding assistant, models like `codellama:34b` or the more recent `qwen2.5-coder:32b` are excellent choices:

```
ollama pull codellama:34b
```

Test the model directly from the command line to ensure everything is working correctly by loading the model and opening an interactive chat session in your terminal:

```
ollama run codellama:34b
```

To complete the private workflow, Cursor must be configured to communicate with the local Ollama server instead of its default cloud-based models. To do so, open Cursor and go to Settings (Ctrl + ,). In the Settings panel, select the *Models* tab. To override the API endpoint, locate the OpenAI API Keys section and click the *Override OpenAI Base URL* option. In the input field that appears, enter the address of your local Ollama server. Crucially, you must append */v1* to the URL to use Ollama's OpenAI-compatible API endpoint (e.g., *http://localhost:11434/v1*). If your Ollama server is running on a different machine on your network, replace *localhost* with that machine's IP address or hostname.

In *Models | Model Names*, click *Add Model* and enter the exact name of the model you pulled with Ollama (e.g., `codellama:34b`). The model is now selectable within Cursor's chat and edit features. For maximum data protection, navigate to *Settings | General* and ensure that *Privacy Mode* is enabled. This step prevents any of your code from being stored on Cursor's servers.

Deploying with Docker

For a more portable and consistent deployment, especially across a team or in different environments, running Ollama within a Docker container is the recommended approach. The `docker-compose.yml` file in Listing 1 defines a service for Ollama, manages data persistence, and includes configuration for GPU access. Table 1 describes the directives. To launch the service, save the file as `docker-compose.yml` and run

```
docker compose up -d
```

in the same directory.

Hardware Strategy

The viability of an on-premises AI workflow hinges on selecting appropriate hardware that can run modern LLMs efficiently without the prohibitive cost of enterprise-grade data center equipment. Careful component selection is key to balancing performance and budget.

The primary hardware constraint for running LLMs is VRAM for GPU-accelerated inference and system RAM for CPU-only inference. General guidelines for a CPU suggest having at least 8GB of RAM for 7B models, 16GB for 13B models, and 32GB for 33B models. However, for a productive developer experience, GPU acceleration is essential. The VRAM requirement is determined by the model's size and its level of quantization. A useful baseline can be established by the popular Llama 3.1 model variants:

- 8B parameter model (4-bit quantization, e.g., q4_K_M): Requires approximately 4.9GB of VRAM, which is well within the capacity of modern consumer GPUs.
- 34B parameter model (4-bit quantization): Requires approximately 20GB of VRAM. This size is a sweet spot for high performance on a single, high-end consumer GPU.
- 70B parameter model (4-bit quantization, e.g., q4_K_M): Requires approximately 43GB of VRAM, which is a critical threshold. It exceeds the capacity of any single consumer GPU, necessitating either a professional-grade card with 48GB of VRAM (e.g., an NVIDIA RTX A6000) or a multi-GPU setup of two consumer cards (e.g., two RTX 4090s with 24GB each).

Table 1: Directives

Directive	Function
`image: ollama/ollama:latest`	Specifies the official Ollama Docker image.
`ports: - "11434:11434"`	Maps port 11434 from the host machine to the container, allowing Cursor to connect to it.
`volumes: - ollama_data:/root/.ollama`	Creates a named Docker volume ollama_data to persist the downloaded models and configuration. This ensures that your models are not lost if the container is restarted.
`deploy`	Grants the container access to the host's NVIDIA GPUs, which requires the NVIDIA Container Toolkit to be installed on the host system. When enabled, Ollama automatically detects and uses the GPU for inference, providing a massive performance boost.
`restart: unless-stopped`	Ensures the Ollama service automatically restarts if it crashes or if the host machine reboots.

For a development team aiming to run models up to the 34B parameter range, a single high-end consumer GPU provides the best balance of cost and performance. Table 2 outlines a sample mid-range, on-premises server build, synthesizing recent component pricing and performance data to provide a realistic capital expenditure estimate.

This build represents a concrete, actionable hardware plan that establishes a clear capital expenditure baseline critical for financial analysis. For a technical decision-maker, this tangible investment figure is a critical data point. The selection of an RTX 4090 is deliberate; its 24GB of VRAM comfortably accommodates the ~20GB requirement for a 34B quantized model, making it an ideal choice for a high-performance, single-GPU developer workstation.

Security Hardening

A privacy-first architecture is incomplete without a robust security posture. Although containing data locally provides a strong foundation, exposing the Ollama API to a network without proper controls introduces significant vulnerabilities. A multilayered defense strategy is required to transform a local setup into a secure, enterprise-ready environment.

By default, the Ollama server is designed for local, single-user access and therefore has no built-in authentication or encryption mechanisms. A common but dangerous practice is to bind the server to 0.0.0.0 to make it accessible to other machines on the network. This action creates an unauthenticated, open API endpoint. Any user or process on the network can then send requests to the Ollama server, potentially leading to resource abuse, unauthorized access to proprietary models, or data leakage if sensitive information is passed in prompts.

The standard industry practice for securing an unauthenticated back-end service is to place a reverse proxy in front of it. The proxy acts as a secure gateway, intercepting all incoming traffic, enforcing security policies, and only forwarding legitimate requests to the Ollama server.

For the highest level of authentication, implementation with NGINX and mutual TLS (mTLS) is recommended. Unlike standard TLS/SSL, where only the server proves its identity to the client, mTLS requires both the client (e.g., the machine running Cursor) and the server (the NGINX proxy) to present valid, trusted certificates to each other

Table 2: Server Build Cost Breakdown (2025)

Component	Model Recommendation	Rationale	Estimated Cost (USD)
GPU	NVIDIA GeForce RTX 4090 (24GB GDDR6X)	The premier consumer GPU for LLM inference, offering the best price-to-performance ratio. Its 24GB VRAM is sufficient for quantized models up to ~34B parameters.	~$1,800 to $2,500
CPU	AMD Ryzen 7 7800X3D	Excellent single-core performance is beneficial for the initial processing of prompts before the task is offloaded to the GPU. During token generation, the CPU's role is minimal if the model fits entirely in VRAM.	~$330
Motherboard	MSI B650 Gaming Plus WiFi	A well-balanced motherboard with support for PCIe 5.0, providing a future-proof path for next-generation GPUs, and four RAM slots for potential memory expansion.	~$170
RAM	64GB (2 x 32GB) DDR5-6000 CL30	Provides ample system memory to prevent the operating system and other applications from becoming a bottleneck, especially when working with large datasets or multiple applications.	~$180
Storage	4TB NVMe PCIe 4.0 SSD	High-speed storage is crucial for minimizing model load times. LLM files in GGUF format can be 20GB or larger, and fast read speeds directly affect how quickly a model can be loaded into VRAM.	~$240
Power supply	1,000W 80+ gold-certified, fully modular	A high-quality power supply unit with sufficient wattage to handle the significant power draw of the RTX 4090 under full load, with headroom for other components.	~$130
Case and cooling	Mid-tower ATX case plus air CPU cooler	A standard chassis and cooling solution ensures adequate airflow to maintain optimal operating temperatures for all components during sustained use.	~$150
Total			~$3,000 to $3,700

Listing 2: NGINX Reverse Proxy with mTLS

```
# /etc/nginx/nginx.conf
events {}
http {
    server {
        listen 443 ssl;
        server_name ollama.internal.corp; # Internal DNS name for the server

        ssl_certificate         /etc/nginx/certs/server.crt;
        ssl_certificate_key     /etc/nginx/certs/server.key;

        # --- Mutual TLS Configuration ---
        ssl_client_certificate  /etc/nginx/certs/ca.crt;
        ssl_verify_client       on; # Enforce client certificate verification
        ssl_verify_depth        1;

        location / {
            # Forward valid requests to the local Ollama server
            proxy_pass http://127.0.0.1:11434;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

before establishing a connection. This arrangement ensures that only authorized and authenticated clients can communicate with the API endpoint.

The `nginx.conf` code in Listing 2 provides a sample configuration for an mTLS-secured reverse proxy for Ollama. To implement it, an internal certificate authority (CA) must be created to issue server and client certificates. The script in Listing 3 demonstrates this process with `openssl`.

The generated `client.key` and `client.crt` would then be securely distributed to developer machines to be used when making API requests.

Overlay Networks

A reverse proxy secures the API endpoint, but it does not solve the problem of how distributed team members can connect to it without exposing the endpoint to the public Internet by port forwarding. The solution is an overlay network that employs Tailscale and WireGuard.

Tailscale is a zero-configuration VPN service built on top of the modern, high-performance WireGuard protocol, creating a private, end-to-end encrypted mesh network (a "Tailnet") that connects an organization's servers, developer machines, and other devices as if they were on the same local network, regardless of their physical location.

Developers running Cursor on their laptops can connect directly to the on-premises Ollama server with the use of its stable, private Tailscale IP address (e.g., 100.x.x.x). The NGINX proxy would be configured to listen on this Tailscale IP address. All traffic is encrypted by WireGuard, and the server's ports are never exposed to the public Internet, effectively eliminating a major attack vector. This approach provides a robust, zero-trust access model that is both secure and simple to manage.

This combination of endpoint authentication and network access control is critical. The community consensus around securing Ollama points to using a reverse proxy, and mTLS provides a much stronger guarantee than simple basic authentication. However, endpoint security alone is insufficient. Overlay networks like Tailscale address the network access challenge, creating a secure, private communication channel over the public Internet. A truly secure on-premises architecture therefore employs both mTLS to authenticate who is allowed to use the API and Tailscale to control who is allowed to even reach the API endpoint. This layered defense strategy is essential for any enterprise-grade deployment.

Analysis and Key Considerations

This section provides a data-driven analysis of the on-premises workflow across the critical dimensions of performance, cost, privacy, and developer experience. Performance for an AI developer tool is a balance between interactive responsiveness (latency) and the ability to handle load (throughput). The on-premises stack is

Listing 3: OpenSSL Certificate Generation

```
#!/bin/bash
CERT_DIR="./certs"
mkdir -p "$CERT_DIR"
# Generate CA Key and Certificate
openssl genrsa -out "$CERT_DIR/ca.key" 4096
openssl req -x509 -new -nodes -key "$CERT_DIR/ca.key" -sha256 -days 3650 -out "$CERT_DIR/ca.crt" -subj "/CN=MyInternalCA"
# Generate Server Key and Certificate, signed by CA
openssl genrsa -out "$CERT_DIR/server.key" 2048
openssl req -new -key "$CERT_DIR/server.key" -out "$CERT_DIR/server.csr" -subj "/CN=ollama.internal.corp"
openssl x509 -req -in "$CERT_DIR/server.csr" -CA "$CERT_DIR/ca.crt" -CAkey "$CERT_DIR/ca.key" -CAcreateserial -out "$CERT_DIR/server.crt" -days 365
# Generate Client Key and Certificate, signed by CA
openssl genrsa -out "$CERT_DIR/client.key" 2048
openssl req -new -key "$CERT_DIR/client.key" -out "$CERT_DIR/client.csr" -subj "/CN=developer-machine"
openssl x509 -req -in "$CERT_DIR/client.csr" -CA "$CERT_DIR/ca.crt" -CAkey "$CERT_DIR/ca.key" -CAcreateserial -out "$CERT_DIR/client.crt" -days 365
```

highly optimized for developer-centric interactive tasks.

To provide a realistic assessment, a standardized benchmark was designed with the following parameters:

- Models: Two popular and powerful code generation models were selected for testing – `codellama:34b` and `qwen2.5-coder:32b` – with the use of the 4-bit quantized GGUF versions (q4_K_M).
- Hardware: The server build is specified in Table 2 and features an NVIDIA GeForce RTX 4090 with 24GB of VRAM.
- Task: A representative code generation prompt was used: *Write a complete Python class for a thread-safe LRU cache using a dictionary and a doubly linked list.*
- Metrics: Time to first token (TTFT), the time elapsed from sending the request to receiving the very first token of the response, is a crucial indicator of perceived latency and responsiveness in an interactive setting; tokens per second (TPS), the total number of tokens generated divided by the total generation time (excluding TTFT) measures the raw throughput of the model.

The on-premises GPU excels at single interactive requests, delivering a highly responsive user experience (Table 3).

The performance of the RTX 4090, with 32-34B parameter models that use Ollama, consistently benchmarks at around 34-37 TPS, which serves as a reliable real-world baseline. The TTFT for the on-premises setup is primarily dictated by the GPU's prompt processing speed, with negligible network overhead.

The on-premises Ollama and RTX 4090 stack provides a superior experience for the individual developer. In the interactive context of a code editor, the most important metric is the perceived responsiveness, which is best measured by TTFT. A developer making a single request for code completion or generation will experience a near-instantaneous response from the local GPU, free of the unavoidable latency of a network round-trip. This workflow is optimized for minimizing interactive latency for a single user.

Cost Projection Analysis

The financial model for the on-premises workflow is a classic capital expenditure (CapEx) model, characterized by a significant upfront investment in hardware. A three-year TCO analysis provides an accurate financial picture of on-premises costs (Table 4).

The financial analysis presents a clear benefit. The on-premises solution is an order of magnitude cheaper over a three-year period compared with cloud alternatives. This dramatic cost difference is a primary driver for organizations to consider the on-premises approach, provided the fixed capacity of the local server can meet their performance and scalability requirements.

Privacy and Compliance

For any organization handling sensitive data, the choice of on-premises infrastructure has profound implications for privacy, data sovereignty, and regulatory compliance.

The on-premises model provides absolute data sovereignty. By design, all data source code, developer prompts, model outputs, and models themselves remain within the organization's physical or logically defined network perimeter. Neither data location nor third-party access is ambiguous; the organization is the sole custodian of the data and is the simplest and most robust privacy guarantee available.

The choice of architecture has a direct effect on the approach to meeting major compliance frameworks like the European Union General Data Protection Regulation (GDPR), the US Health Insurance Portability and Accountability Act (HIPAA), and the System and Organization Controls (SOC 2). The on-premises model simplifies compliance from a data residency perspective but places the

Table 3: Benchmark Results

Metric	Model	Ollama and RTX 4090
Throughput (TPS)	`codellama:34b`	~36 tokens/sec
	`qwen2.5-coder:32b`	~34 tokens/sec
Latency (TTFT)	`codellama:34b`	~150ms

Table 4: Three-Year TCO

Cost Component	Ollama+Cursor (on-premises)	Explanation
Hardware/Compute	$3,700 (upfront CapEx)	The primary cost is the server itself. The upfront cost was calculated with the high-end estimate from the build in Table 2.
Storage	Included in hardware cost	–
Data transfer	$0	Internal network
Power and cooling	~$714	This ongoing cost is calculated on the basis of the server's power consumption assuming the 1,000W power supply runs at an average 70-percent load during a standard eight-hour workday for 250 workdays a year at the average US electricity rate of $0.17/kWh.*
Maintenance	Qualitative (engineer time)	Although no direct fees are charged, the cost of server setup, updates, and maintenance is a small fraction of an engineering or IT team's time.
Three-year TCO	~$4,414	

* (1.0kW x 0.70 load x 8hr/day x 250 days/yr x $0.17/kWh) x 3yr=$714

entire burden of implementing and proving all other security controls on the organization. With GDPR, for example, data is inherently stored in the location of the server. For HIPAA and SOC 2, the organization is solely responsible for implementing all required safeguards on its own infrastructure, and the system would be part of the organization's overall audit scope.

Developer Experience and MLOps Maturity

The day-to-day workflow, from experimenting with new models to deployment and monitoring, is a key strength of the Ollama stack.

Ollama offers extremely high flexibility. The open source community is constantly releasing new and improved models, often in the GGUF format. With Ollama, a developer can download and begin experimenting with a new state-of-the-art model with a single `ollama pull` command, often within minutes of its release, which enables a culture of rapid iteration and allows the team to always have access to the latest advancements.

The on-premises stack relies on the broader open source ecosystem. For monitoring, a common solution is to use a proxy to export metrics to Prometheus, which are then visualized in Grafana. For continuous integration and continuous delivery (CI/CD), standard tools like GitHub Actions or CircleCI can be used to build and test custom models defined in an Ollama Modelfile. This approach offers maximum flexibility but requires the team to integrate and maintain these disparate tools.

A modern machine learning (ML) Ops pipeline for Ollama would follow GitOps principles. The Modelfile, which defines a custom model configuration, would be stored and version-controlled in a Git repository. Merging a change to the main branch would trigger a CI/CD pipeline. This pipeline could build a new Docker image with the updated model baked in, run a suite of automated tests against it, and, if successful, push the new image to a container registry and deploy it to the on-premises server. Monitoring would involve an external tool scraping a metrics endpoint for performance data.

Conclusion

The strategic decision to adopt an on-premises Ollama workflow prioritizes data sovereignty, cost control, and developer agility. Organizations should choose the Ollama + Cursor on-premises workflow for:

- Privacy and sovereignty if your number one priority is absolute data control and you have a strict policy against proprietary code or sensitive data leaving your network perimeter.
- Scalability and performance if your primary use case is empowering a small-to-medium-sized development team, for whom interactive low-latency responses are more critical than handling massive concurrent load.
- Budget model if you prefer a predictable, one-time capital expenditure (CapEx), and your primary goal is to achieve the lowest possible TCO over the long term.
- Technical expertise and MLOps if you have strong in-house expertise in hardware management, network security, and systems administration, and you prefer the flexibility of building a bespoke MLOps pipeline with open source tools.

Developer agility and experimentation if your team's workflow requires the ability to test, fine-tune, and deploy the latest open source models rapidly and with minimal friction and turnaround time.

This comprehensive analysis reveals a clear use case for an on-premises AI developer workflow. The Ollama + Cursor on-premises workflow offers an unmatched solution for privacy, control, and cost-effectiveness. For individual developers, startups, and enterprises where intellectual property protection is paramount, this stack provides a secure and highly agile environment for internal development. Its primary limitations are the fixed capacity of the hardware and the operational responsibility placed on the organization.

Looking forward, the landscape will continue to evolve rapidly. The performance of consumer-grade hardware is increasing, and open source models are becoming more powerful and efficient, strengthening the case for local, on-premises solutions. The fundamental choice for a workflow optimized for private, agile development will likely remain a key strategic option for many organizations. ■

Info

[1] Ollama: [https://ollama.com]
[2] Cursor: [https://cursor.com]

The Author

Dr. B. Thangaraju is a Professor at the International Institute of Information Technology Bangalore (IIITB), where he leads the Open Source Technology Lab. He has previously served as a Research Associate at the Indian Institute of Science (IISc.) and as a Senior Consultant at Wipro Technologies. His areas of expertise include Linux kernel development, embedded and real-time systems, DevOps, container technologies, and OpenStack, and he is the author of the book *Real-World DevOps Practices: A Case Study Approach* (Wiley, 2024).

Vidhu Arora is a DevOps practitioner with a solid foundation in system programming and automation. He has built and deployed scalable backend systems and brings hands-on experience with infrastructure tools like Docker, Jenkins, and Kubernetes.

Shreyan Gupta holds a degree in Computer Science and Engineering from IIIT Bangalore. He is deeply passionate about blockchain technology and has collaborated with law enforcement agencies on transnational investigations involving more than $85 million.

Hone Your Skills
– with –
Special Issues!

GETTING STARTED WITH LINUX
Cool Linux Hacks
LINUX SHELL HANDBOOK
LibreOffice Expert
THE BEST OF LINUX MAGAZINE

Get to know Shell, LibreOffice, Linux, and more from our Special Issues library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

MakerSpace
MakerSpace #02
MakerSpace #03
MakerSpace #04
MakerSpace #05

background image © roystudio, 123RF.com

Check out the full library!

shop.linuxnewmedia.com/shop

Signing software artifacts

Sign Here

This tutorial on deploying Sigstore (Cosign, Rekor, Fulcio) for container and binary integrity on Ubuntu covers keyless compared with key-based models, Fulcio's OIDC trust, Rekor transparency usage, and verification policy design. By Marcin Gastol

Sigstore [1] is an open source solution for cryptographically signing software artifacts and verifying those signatures in a transparent, tamper-evident manner. Created to improve software supply chain security by making signatures easy to use and difficult to subvert, the Sigstore ecosystem comprises three core components: Cosign, Fulcio, and Rekor. Cosign [2] is the command-line tool that developers and CI/CD systems use to sign container images and other artifacts and to verify those signatures [3]. Fulcio is a lightweight certificate authority that issues short-lived signing certificates with the use of OpenID Connect (OIDC) identities. Rekor is a public transparency log that records metadata about each signing event in an immutable ledger. Together, these components let anyone verify who signed a piece of software and ensure that the signature is recorded on a public log, deterring and detecting tampering or mischief.

This integration of Fulcio (for identity verification) and Rekor (for public logging) with the Cosign command-line interface (CLI) provides a modern approach to trust in software artifacts. Importantly, Cosign stores signatures as Open Container Initiative (OCI) artifacts in container registries next to the images themselves, so distributing and retrieving signatures is as straightforward as pulling container images.

Setting Up Cosign

Getting Cosign up and running on an Ubuntu system is straightforward. Because Cosign is distributed as a single binary, installation is by package managers or direct download. In recent Ubuntu releases, Cosign is available in the default repositories (universe), so you can install it with Apt:

```
sudo apt install cosign
```

This command places the `cosign` executable on your system. Alternatively, you can download the latest binary from the Sigstore GitHub release page or use other package tools [4]. The binary is relatively small and has no heavy dependencies, which makes deployment easy. Once installed, you can verify by running

```
cosign version
```

to confirm you have a recent release; newer versions include important stability and security improvements. Before using Cosign, ensure your Ubuntu environment itself is secure and up to date by applying the latest security patches,

```
apt update && apt upgrade
```

and configuring basic security measures. For instance, use SSH key authentication (disable password login) for remote access and enable

Lead Image © olegdudko, 123rf.com

the UFW firewall to allow only necessary ports (e.g., SSH on 22, and perhaps 443 if needed for any API calls) while blocking others. These measures aren't specific to Cosign, but they ensure that the machine you're using for signing and verifying artifacts isn't itself easily compromised. Remember that signing keys or credentials will reside on this system (even if briefly), so its security is paramount.

One of Sigstore Cosign's distinguishing features is its keyless signing capability, which streamlines the process by doing away with manual key management. In the next sections, I unpack how keyless signing works and why it's beneficial, then I compare it with the traditional key-based approach.

Keyless Signing

In keyless mode, you do not provide Cosign with any pre-existing private key. Instead, Cosign generates an ephemeral key pair on the fly and gets a signing certificate for the public key from Fulcio after confirming your identity. The workflow for a typical keyless signing goes like this: You run a `cosign sign` command without specifying a key, and Cosign reaches out to Fulcio to request a certificate. Fulcio prompts you to authenticate with an OpenID Connect provider. For the public instance of Fulcio, this can be GitHub, Google, or Microsoft accounts.

For an interactive user, Cosign usually opens a browser for login or provides a URL/code if on a headless server. Once you log in successfully and Fulcio verifies your identity (e.g., you are *alice@example.com* in Google), Fulcio issues a short-lived X.509 certificate, binding that identity to the newly generated public key. Cosign then uses the corresponding ephemeral private key (which exists only in memory) to sign the artifact. The signature, along with the certificate and an inclusion proof of the Rekor log entry, can be saved as a bundle or attached to the artifact. Cosign immediately throws away the ephemeral private key after use. The whole signing event (artifact digest, signature, certificate) is also sent to Rekor to be recorded in the transparency log.

The security advantages of keyless signing are significant. Because the private key exists only momentarily and is unique to that signing session, there's no long-lived key for an attacker to steal. Even if someone could grab the in-memory key, it would be useless almost immediately after, because the certificate binding it to your identity expires and the key won't be reused. In traditional signing, a leaked private key can be catastrophic (attackers could sign malware with it at any time); keyless signing largely mitigates that risk by design. Additionally, the signed artifact carries a cryptographic chain of evidence about who signed it – the certificate from Fulcio includes your identity, and verification can confirm that identity was used and trusted by Fulcio.

Key-Based Signing

In contrast to keyless signing, Cosign also supports the traditional model of signing with a user-managed key pair, which might be necessary in air-gapped environments, in organizations with an existing public key infrastructure (PKI), or simply by personal preference. The use of key pairs with Cosign involves generating or obtaining a signing key, keeping the private part safe, and providing it to Cosign when signing. Cosign can generate an elliptic curve digital signature algorithm (ECDSA) key pair,

```
cosign generate-key-pair
```

which produces an encrypted private key file (password-protected by strong encryption) and a public key file. You could also use any existing ECDSA key or even an RSA key if you convert it appropriately.

Use of your own key pair puts the onus on you to manage that key's security. You should store the private key in a secure vault or hardware device, if possible, and limit access to it. Cosign integrates with hardware tokens (e.g., YubiKeys or hardware security modules (HSMs) and cloud key management service (KMS) providers so that you might not ever even handle the raw key material. Cosign supports keys in HashiCorp Vault, Google Cloud KMS, AWS KMS, Azure Key Vault, and others with the use of provider plugins. When Cosign needs to sign, it calls the KMS, which performs the signature operation inside a secure environment. Many enterprise setups prefer this solution because it centralizes key control and usage logging.

Signing Software Artifacts

The most common use case is signing container images, so that will be the primary example; however, note that Cosign can sign many other artifacts in a similar fashion (files, binaries, packages, etc.). The process involves identifying the artifact to sign, running the Cosign `sign` command (with or without a key, depending on mode), and verifying that the signature and log entry were recorded.

Before running the `sign` command, determine the reference for the artifact. Cosign allows signing a container image by tag (e.g., `myapp:latest`), but this method is strongly discouraged in favor of signing by immutable digest because a tag like *latest* can be moved to point to different images over time. You wouldn't want to sign an image, then have the tag point to something else that isn't signed. Instead, use the image's digest, which uniquely identifies its content. If you're using Docker or another tool to push images, you can usually get the digest after

```
docker push
```

which outputs the digest, or you can run

```
docker inspect
```

Cosign also has a subcommand to resolve a tag to digest,

```
cosign triangulate
```

although it's not strictly necessary if you already have the digest.

Verifying Signatures

Signing is only half of the equation: You gain security when you verify those signatures before using or trusting an artifact. Cosign makes verification as simple as signing, but you need to understand some important details, especially with the introduction of certificate identities and the transparency log.

When you verify a signed artifact with Cosign, the tool fetches the signature (from the registry or from a file you provide), obtains any needed certificate or public key info (either from the registry attachment, Rekor, or from flags you pass), and then performs a series of checks:

- **Cryptographic signature check.** The signature is verified to be mathematically valid for the given artifact (image digest or file hash) with the corresponding public key.
- **Certificate check (if keyless).** If the signature was done keylessly, Cosign also verifies the signing certificate, checking that the certificate is issued by Fulcio's root (meaning it's a real Sigstore cert) and is not expired and validating the certificate's signature.
- **Identity check.** You must specify which identities you trust.
- **Transparency log check.** Cosign verifies the Rekor transparency log entry for this signature. If you have the signature bundle, Cosign uses it to confirm the entry exists in the log. If a bundle isn't provided, Cosign by default queries the Rekor server for an entry matching the artifact and signature to get the proof. It then checks the log's own signature against Rekor's public key to ensure the entry is authentic.
- **Time/expiry check.** Another subtle check Cosign performs is ensuring that the signature's timestamp falls within the validity window of the signing certificate (if keyless). Because Fulcio certs are short-lived, Cosign wants to ensure the signature was made while the cert was valid and not, say, after it expired, which could hint at someone trying to reuse a cert illegitimately. When Cosign queries Rekor and gets the entry, it sees the integrated timestamp of when the log stored it. It compares that against the certificate's validity period to make sure it aligns. If something was signed outside the cert's lifetime, that's a red flag, and Cosign would fail verification. This check enforces that the ephemeral nature of keyless certs is honored.

When you run `cosign verify`, you'll see output reflecting these checks. A successful verification for an image shows a JSON payload of the signature's details.

Conclusion

The threat of software supply chain attacks is real and growing, and signing artifacts is becoming a recommended or required practice. Cosign and Sigstore bring what used to be complex PKI machinery down to earth for everyday use, with no need to manage keys and distribute certs or fingerprints manually. Many of these tasks are handled for you in an open, auditable way. With the use of Cosign on any Linux distribution, integrated with cloud services and container registries, you implement state-of-the-art security in a way that scales with infrastructure.

The configuration and optimization tips provided here aim to ensure you get the maximum security benefit with minimal friction. As you deploy Cosign in your environment, always keep both the theoretical (why each step matters for trust) and practical (how to implement it correctly on your systems) knowledge in mind. This combination will enable you to troubleshoot issues, tailor the setup to your needs, and advocate for secure practices within your team.

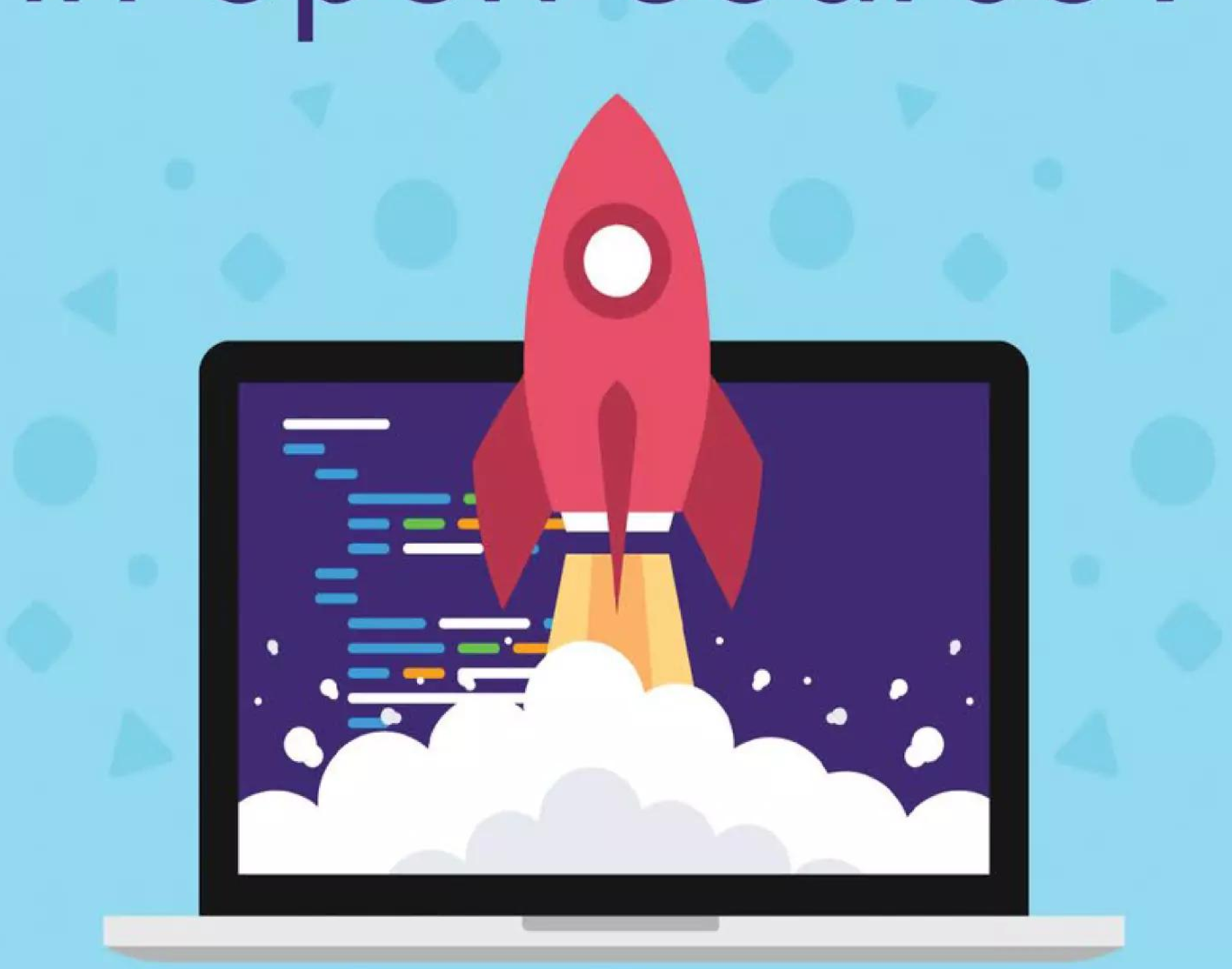
Info

[1] Sigstore Cosign: [https://github.com/sigstore/cosign]
[2] Sigstore Blog: [https://blog.sigstore.dev/]
[3] Official Sigstore documentation: [https://docs.sigstore.dev/]
[4] Cosign install: [https://docs.sigstore.dev/cosign/system_config/installation/]

The Author

Marcin Gastol is a Senior DevOps Engineer and a Microsoft MVP and Certified Trainer with extensive experience in Azure technologies. He has taught various IT subjects, hosts a blog that addresses multiple IT areas [https://marcingastol.com/], and speaks at various conferences.

Looking for your place in open source?



Set up job alerts and get started today!

OpenSource
JOB HUB

opensourcejobhub.com

@opensourcejobhub.com

Open Source JobHub

Open Source JobHub

@osjobhub@fosstodon.org

Photo by NASA Hubble Space Telescope on Unsplash

Resurrecting bWatch

Bring It Back

After 21 years, Jeff returns to bWatch, a simple cluster monitoring tool from the early days of Beowulf. By Jeff Layton

When the world was dominated by dinosaurs, a new beast arose from the depths: one that relied on the Earth's community to grow and thrive. It was Beowulf. No, not the Scandinavian warrior, but an approach to high-performance computing (HPC) that uses common x86 processors, conventional Ethernet networking, the Message Passing Interface (MPI) or a parallel virtual machine (PVM), and Linux.

With Beowulf, everything was new. Previous HPC systems all had proprietary tools to manage and monitor servers (nodes) in the cluster, so the tools for Beowulf clusters had to be developed, including those to monitor clusters. During the first few years of this Clusterian period of HPC, one of the key requirements was a simple tool to monitor nodes – something simple that would give the status of all nodes in the cluster.

A visual presentation of the load on the nodes (with the `uptime` command), node uptime, and maybe even some memory usage would let you determine the status of a cluster with a quick glance at the screen. At that time, lots of people were experimenting with and developing clusters, so many were homemade and small: A 60-node cluster was considered large, so you would easily be able to see all of the nodes on one screen with a little scrolling.

bWatch

Python was still in early days, but Tcl/Tk was very popular. The high-level, general-purpose, interpreted and dynamic Tool Command Language (Tcl) [1] was created by John Ousterhout in the 1990s. From the name you can probably tell that it was intended to be a language for writing tools for whatever platform you were using. (I will begrudgingly admit there are more platforms than Linux and other Unix-like operating systems.) Because it was interpreted, you didn't have to change code and recompile constantly. You could just change the code and run.

Ousterhout also needed a tool for developing graphical user interfaces (GUIs) for Tcl applications, so about one year after starting Tcl, he created Tk [2], which has expanded beyond just Tcl with bindings to many other languages. Tk provides widgets commonly used for desktop applications such as buttons, menus, a canvas, text, labels, and so on.

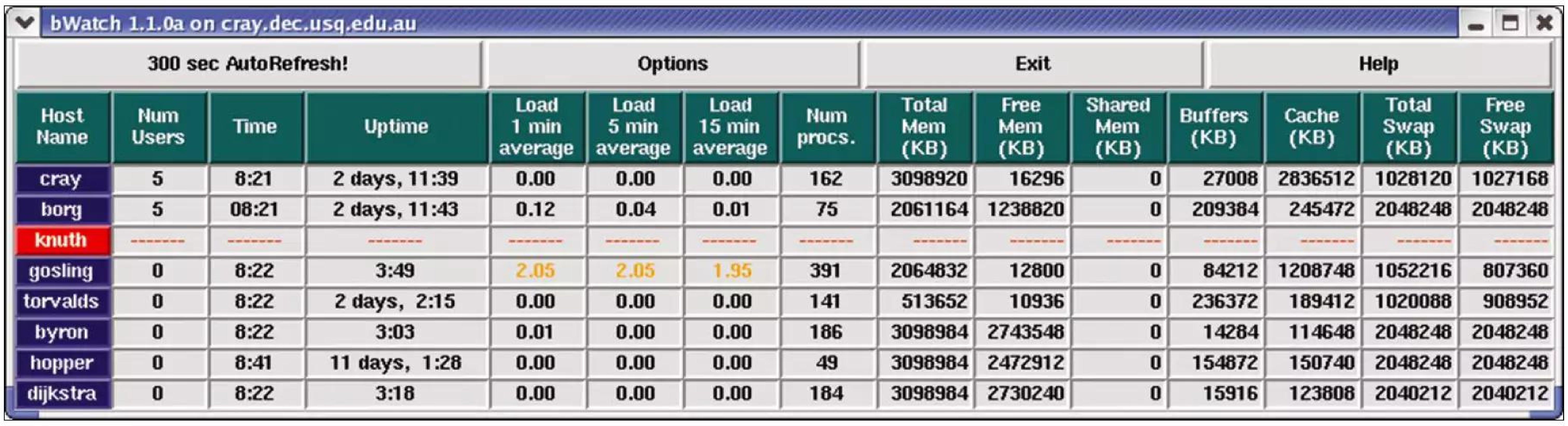
To create something useful for Beowulf clusters, Jacek Radajewski used these two tools to write a simple monitoring tool, bWatch [3]. Although simple in concept, when first released, a huge portion of people using and developing Beowulf clusters used it. The first time I used it, I was very excited because I could get a visual representation of the state of the cluster (Figure 1), rather than having to log in to each node or use a command-line tool to run parallel commands on all the nodes and parse through the data.

bWatch concepts are pretty simple: The idea is to gather information about the node by opening a shell on that node and running Linux commands. Originally, bWatch used the remote shell (`rsh`) [4] to execute shell commands on another node. Although `rsh` is very insecure (it transmits everything without encryption), at the time, it was commonly used, so bWatch used it.

bWatch shells to the node and runs the `uptime` command to find the current time on the node, the number of users, how long the node has been up, and the 1-, 5-, and 15-minute load averages on the node. Isn't it amazing how much information is contained in this one command?

bWatch also runs the command

```
cat /proc/meminfo
```



bWatch 1.1.0a on cray.dec.usq.edu.au

Host Name	Num Users	Time	Uptime	Load 1 min average	Load 5 min average	Load 15 min average	Num procs.	Total Mem (KB)	Free Mem (KB)	Shared Mem (KB)	Buffers (KB)	Cache (KB)	Total Swap (KB)	Free Swap (KB)
cray	5	8:21	2 days, 11:39	0.00	0.00	0.00	162	3098920	16296	0	27008	2836512	1028120	1027168
borg	5	08:21	2 days, 11:43	0.12	0.04	0.01	75	2061164	1238820	0	209384	245472	2048248	2048248
knuth	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------
gosling	0	8:22	3:49	2.05	2.05	1.95	391	2064832	12800	0	84212	1208748	1052216	807360
torvalds	0	8:22	2 days, 2:15	0.00	0.00	0.00	141	513652	10936	0	236372	189412	1020088	908952
byron	0	8:22	3:03	0.01	0.00	0.00	186	3098984	2743548	0	14284	114648	2048248	2048248
hopper	0	8:41	11 days, 1:28	0.00	0.00	0.00	49	3098984	2472912	0	154872	150740	2048248	2048248
dijkstra	0	8:22	3:18	0.00	0.00	0.00	184	3098984	2730240	0	15916	123808	2040212	2040212

Figure 1: **Sample screenshot of bWatch (© Jacek Radajewski).**

to gather information about the memory being used. With the output from just this command and `uptime`, you can get an idea of the status of the node.

Running bWatch

One recent morning, I saw the word "bWatch" on a dry erase board in my home office. Evidently at some point in the last five years or so, I wrote that, but I couldn't remember exactly why. Regardless, that morning I decided to try to get bWatch running again on today's Linux and hardware – or at least give it a good try, even though my Tcl knowledge is gone.

The last version of bWatch [5] posted is 1.1.0a with a date stamp of June 15, 2004, which makes it about 21 years old now. There's no chance this will work, right? I wanted to try it anyway, but I will probably have to make some changes along the way.

wishx and ssh

bWatch is written in Tcl, which I didn't remember at all, but it's fairly easy to read. The first line in the `bWatch.tcl` code is the shebang (`#!/bin/<something>`) that points to the command or tool the code is supposed to run. In the case of bWatch, it is `#!/user/bin/wishx` (Windowing Shell), a TCL and Tk interpreter. However, `wishx` has since given way to just plain `wish`, which I easily installed on my Ubuntu 24.04 system; then, I changed the interpreter from `wishx` to `wish` on the first line of the code. Easy enough.

bWatch uses `rsh`, so I knew I had to change that, as well. I searched through the code and `rsh` is defined in the line,

```
set command(rsh) "/usr/bin/rsh"
```

so I changed the `set command` target to `ssh`:

```
set command(rsh) "/usr/bin/ssh"
```

Everywhere else in the code, `rsh` itself is not used, only `set(command)`.

.bWatchrc.tcl File

The next issue was how to specify the node names to be used, which meant I had to check the `README` file, where I found that the list of nodes (hosts) is specified in a file named `.bWatchrc.tcl` in the root directory:

```
set listOfHosts {node1 node2 node3 ➘
  node4 node5 node6 node7 node8}
```

I'm not sure what restrictions are imposed on node names, so I just used the output from the command `hostname` on all nodes. (I only used one node for this article.)

First Run

To install bWatch, use:

```
sudo make install
```

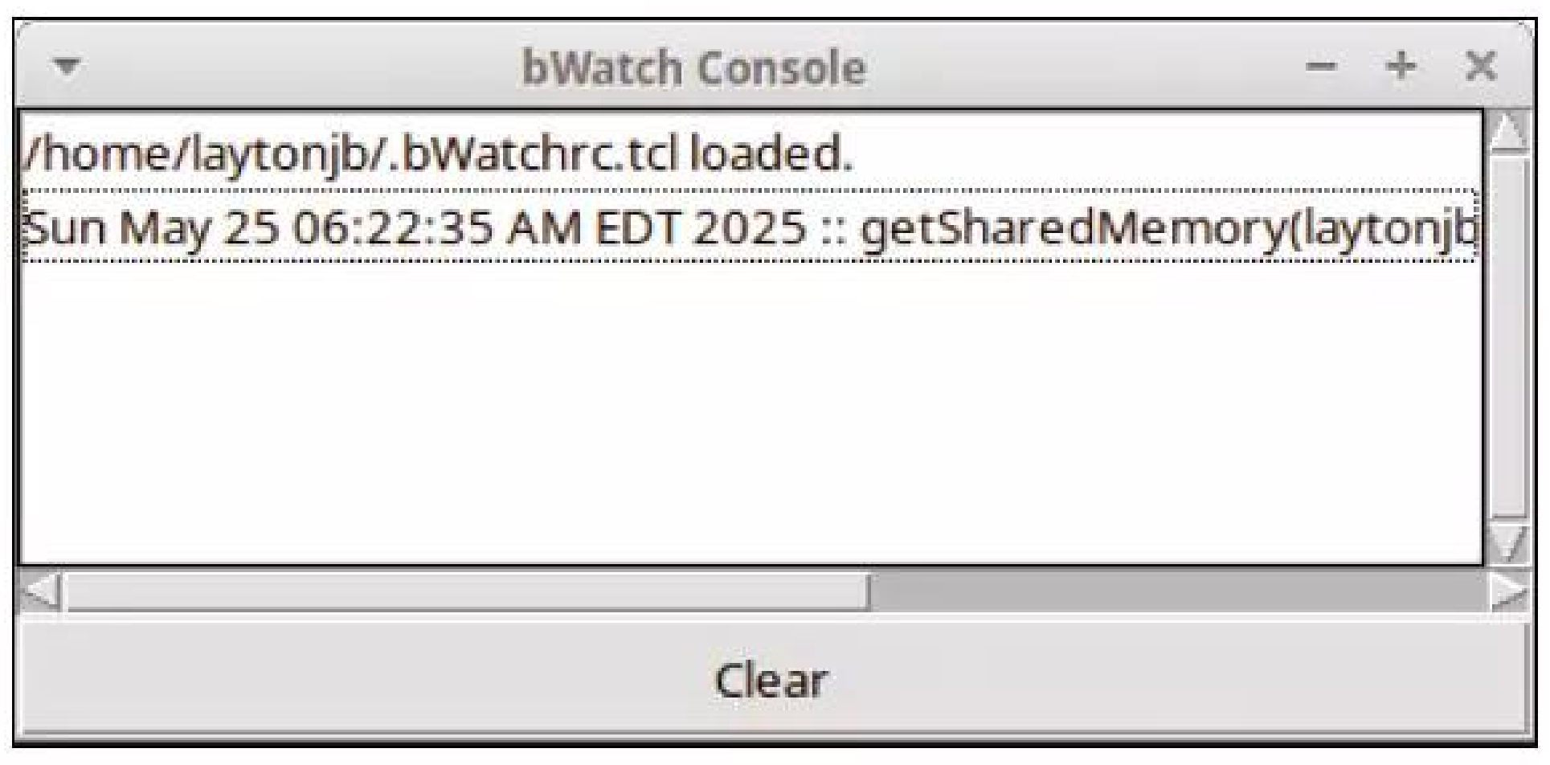
This command puts the code in `/usr/local/bin`, which is also in my default `$PATH`; however, you might check that it is there. Next, I ran the command `bWatch.tcl` (Figure 2). Everything sort of looks good, but notice that the `Shared Mem` column has no values. Something is amiss.

When I ran bWatch, a `wish` console window popped up (Figure 3). The complete error message reads,

```
getSharedMemory(laytonjb-Precision-7680):➘
  child process exited abnormally
```

so something is going on with the `getSharedMemory` procedure.

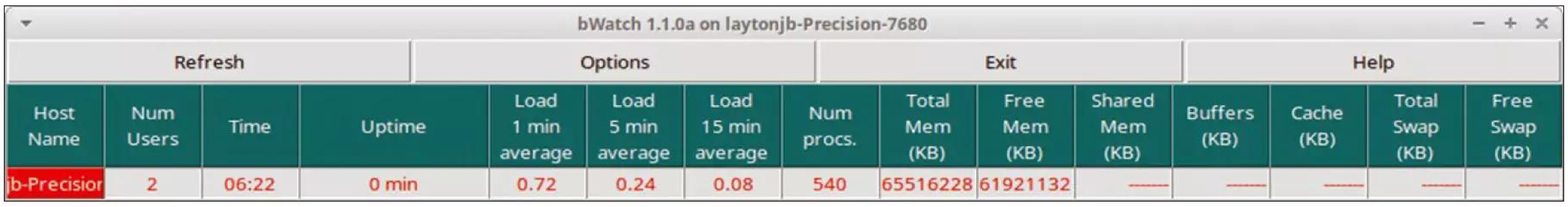
Before digging into this error, I tried turning off data collection for `getSharedMemory` by clicking on the *Options* button, deselecting the *Shared Memory* option (Figure 4), then clicking *Apply* and *OK* to close the window.



bWatch Console

```
/home/laytonjb/.bWatchrc.tcl loaded.
Sun May 25 06:22:35 AM EDT 2025 :: getSharedMemory(laytonjb
```

Figure 3: **A wish console message.**



bWatch 1.1.0a on laytonjb-Precision-7680

Host Name	Num Users	Time	Uptime	Load 1 min average	Load 5 min average	Load 15 min average	Num procs.	Total Mem (KB)	Free Mem (KB)	Shared Mem (KB)	Buffers (KB)	Cache (KB)	Total Swap (KB)	Free Swap (KB)
jb-Precisior	2	06:22	0 min	0.72	0.24	0.08	540	65516228	61921132	------	------	------	------	------

Figure 2: **First run of bWatch in 21 years on my systems.**

At the top of the main bWatch window, selecting *Refresh* causes bWatch to update the information for all nodes possible (Figure 5).

Back to Shared Memory

Although missing just the shared memory information is probably not a big deal, it would be nice to get the data originally captured in bWatch. The procedure that gets the shared memory information in `bWatch.tcl` is shown in Listing 1.

Notice that Listing 1 runs the command `grep MemShared` from the information gathered with `cat /proc/meminfo`. (You need to dig into the code a bit to find out that it stores the result from `cat` so it can be searched for the needed information.)

Listing 1: Getting Shared Memory

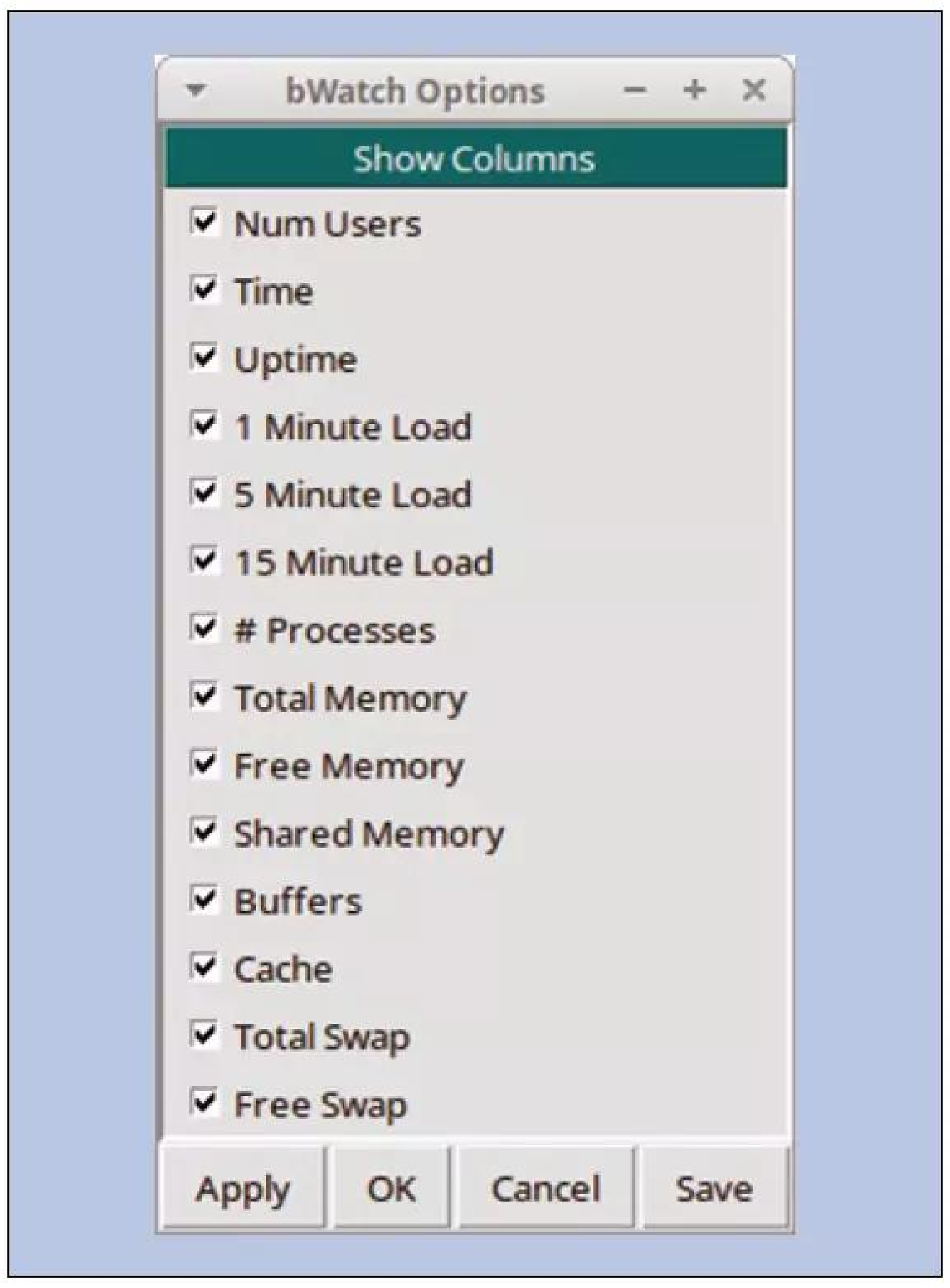
```
proc getSharedMemory {host} {

global command bWatchDir

set errorCode \
[catch {set sharedMemory \
[exec cat $bWatchDir/bWatchMemInfo | grep MemShared]}
   errorMessage]
# check if the command returned an error
if {$errorCode} {
set date [getDate]
.console.messageList insert end \
"$date :: getSharedMemory($host) : $errorMessage"
return "error"
} else {
return $sharedMemory
}
```

The output of `cat /proc/meminfo` from a modern Linux system won't show `MemShared`. The closest output is `Shmem`, so you can just change the `grep` command to `grep Shmem`. If you do this and restart `bWatch.tcl`, you'll see that the *Shared Mem* column reappears (Figure 6). Note that you might have to go back and re-select *Shared Memory* under *Options* to get it to show.

Summary

I remember using bWatch many years ago on my clusters, and I loved the simplicity of the GUI and its ease of configuration, along with the simple information it provided. After 21 years, I was surprised how easily I got bWatch working again. I think that shows the importance of keeping programming languages and GUIs somewhat fixed, along with Linux interfaces (e.g., `/proc/meminfo`), with a few small exceptions. I also like that you don't have to be an admin to use bWatch. Even a humble user can gather this data.

Give bWatch a whirl and poke around with some of the options (which you can save to your `.bWatchrc.tcl` file). Although there really is no need, I think it would be interesting to port bWatch to Python, because it would probably be faster than I relearning Tcl/Tk. ■



☑ Num Users
☑ Time
☑ Uptime
☑ 1 Minute Load
☑ 5 Minute Load
☑ 15 Minute Load
☑ # Processes
☑ Total Memory
☑ Free Memory
☑ Shared Memory
☑ Buffers
☑ Cache
☑ Total Swap
☑ Free Swap

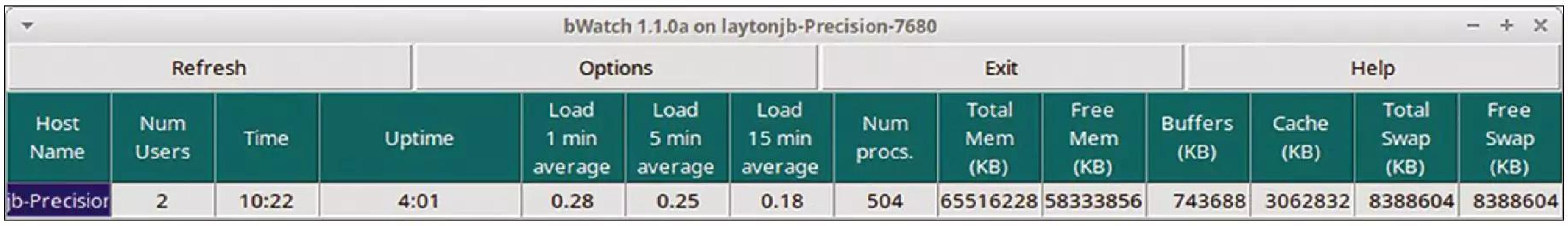
Figure 4: Options selection box.

Info

[1] Tool Command Language: [https://en.wikipedia.org/wiki/Tcl]
[2] Tk: [https://en.wikipedia.org/wiki/Tk_(software)]
[3] bWatch: [https://bwatch.sourceforge.net/]
[4] rsh: [https://en.wikipedia.org/wiki/Remote_Shell]
[5] Latest version of bWatch: [https://bwatch.sourceforge.net/]

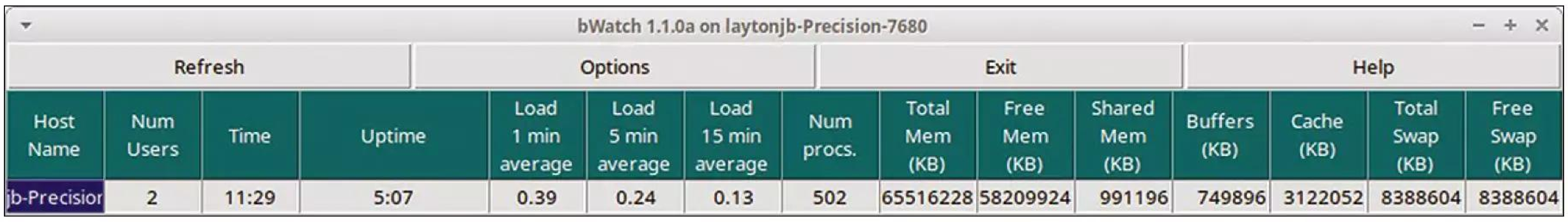
The Author

Jeff Layton has been in the HPC business for over 30 years (starting when he was 4 years old). When he's not grappling with a stubborn systemd script, he's looking for deals for his home cluster. His twitter handle is @JeffdotLayton.



Host Name	Num Users	Time	Uptime	Load 1 min average	Load 5 min average	Load 15 min average	Num procs.	Total Mem (KB)	Free Mem (KB)	Buffers (KB)	Cache (KB)	Total Swap (KB)	Free Swap (KB)
jb-Precisior	2	10:22	4:01	0.28	0.25	0.18	504	65516228	58333856	743688	3062832	8388604	8388604

Figure 5: Refreshed bWatch.



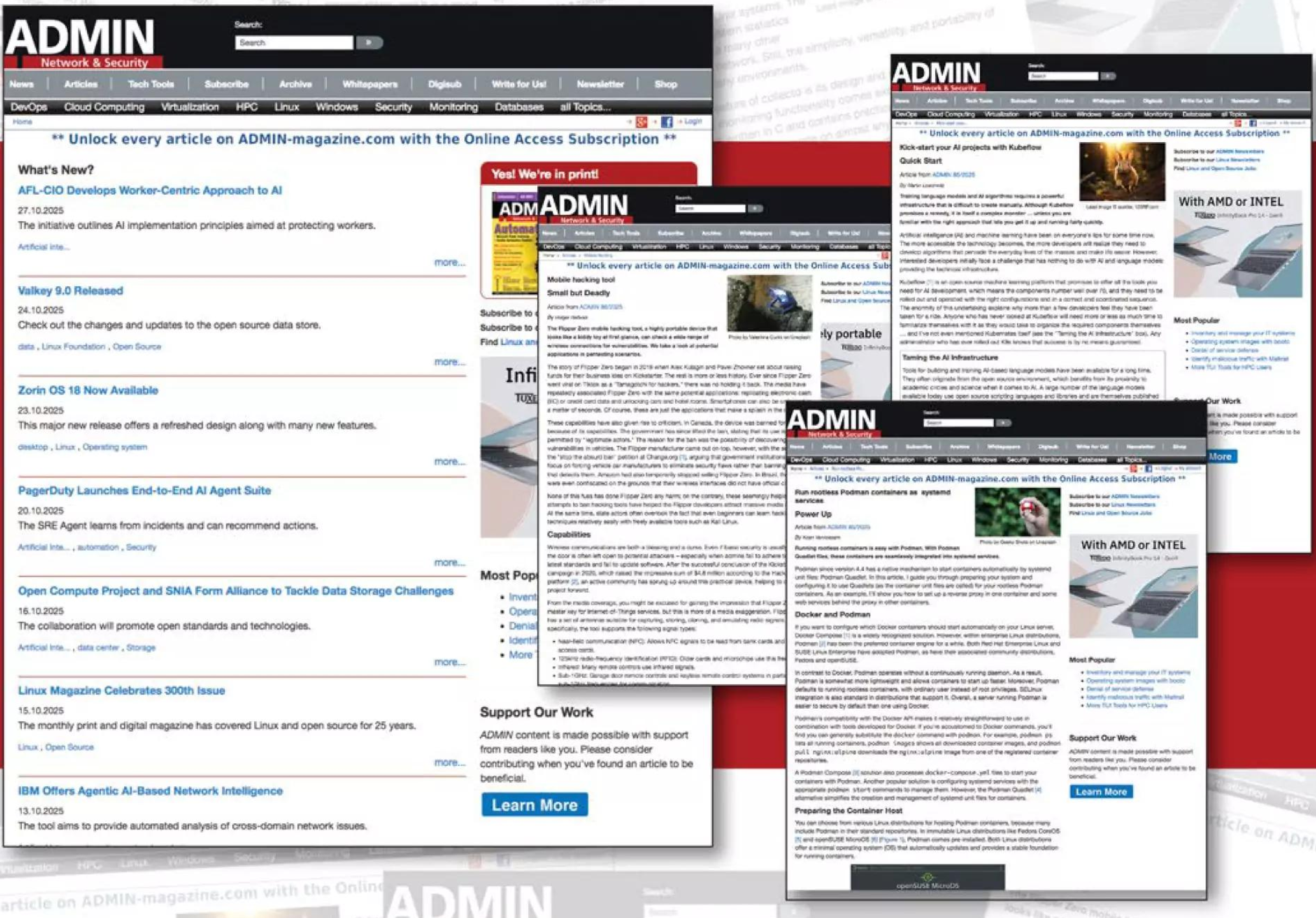
Host Name	Num Users	Time	Uptime	Load 1 min average	Load 5 min average	Load 15 min average	Num procs.	Total Mem (KB)	Free Mem (KB)	Shared Mem (KB)	Buffers (KB)	Cache (KB)	Total Swap (KB)	Free Swap (KB)
jb-Precisior	2	11:29	5:07	0.39	0.24	0.13	502	65516228	58209924	991196	749896	3122052	8388604	8388604

Figure 6: The reappearance of Shared Mem.

■

Get Access to Every Article on ▸ADMIN-magazine.com◂

This monthly subscription gives you access to the full ADMIN website, including thousands of technical articles, news, features, and online content back to 2013.


ADMIN
Network & Security
Search:
News
Articles
Tech Tools
Subscribe
Archive
Whitepapers
Digisub
Write for Us!
Newsletter
Shop
DevOps
Cloud Computing
Virtualization
HPC
Linux
Windows
Security
Monitoring
Databases
all Topics...
Home
Login
Learn More

** Unlock every article on ADMIN-magazine.com with the Online Access Subscription **

What's New?

AFL-CIO Develops Worker-Centric Approach to AI
27.10.2025
The initiative outlines AI implementation principles aimed at protecting workers.
Artificial Inte...
more...

Valkey 9.0 Released
24.10.2025
Check out the changes and updates to the open source data store.
data , Linux Foundation , Open Source
more...

Zorin OS 18 Now Available
23.10.2025
This major new release offers a refreshed design along with many new features.
desktop , Linux , Operating system
more...

PagerDuty Launches End-to-End AI Agent Suite
20.10.2025
The SRE Agent learns from incidents and can recommend actions.
Artificial Inte... , automation , Security
more...

Open Compute Project and SNIA Form Alliance to Tackle Data Storage Challenges
16.10.2025
The collaboration will promote open standards and technologies.
Artificial Inte... , data center , Storage
more...

Linux Magazine Celebrates 300th Issue
15.10.2025
The monthly print and digital magazine has covered Linux and open source for 25 years.
Linux , Open Source
more...

IBM Offers Agentic AI-Based Network Intelligence
13.10.2025
The tool aims to provide automated analysis of cross-domain network issues.

Yes! We're in print!

Most Popular

Support Our Work

ADMIN content is made possible with support from readers like you. Please consider contributing when you've found an article to be beneficial.

Mobile hacking tool
Small but Deadly

Kick-start your AI projects with Kubeflow
Quick Start

Run rootless Podman containers as systemd services
Power Up

Activate your access today, and start reading our online archive!

Customers in Europe and Switzerland (EUR/CHF):
Please contact subs@admin-magazine.com

Customers in the United Kingdom (GBP)

Customers in all other countries (USD)

VM isolation for secure containerized workloads

The Best of Both Worlds

We explore the architecture and configuration of Kata Containers on Ubuntu with Docker, focusing on how lightweight virtual machines can deliver strong workload isolation while preserving the agility and performance of traditional containers. By Marcin Gastol

Running applications in containers is popular for its efficiency and speed, but security remains a concern in multitenant or untrusted workloads. The Kata Containers technology addresses this weakness by running containers inside lightweight virtual machines (VMs) to provide an extra layer of isolation **[1]**.

In essence, Kata Containers merges the speed and simplicity of containers with the strong isolation traditionally provided by VMs, giving each container its own stripped-down VM complete with a dedicated kernel and hardware-enforced isolation and significantly reducing the risk of container escapes affecting the host or other workloads. The result is a container runtime that feels like a standard container from a user perspective but that delivers enhanced security akin to a VM sandbox. This approach has been adopted in cloud environments and production systems where security is important.

Kata Containers and VM Isolation

Traditional containers use runtimes like runC and share the host's kernel.

Processes inside a container directly use the host Linux kernel, which can be a security concern if a containerized process manages to break out or exploit the kernel. Kata Containers takes a different approach: When you launch a container with Kata, it boots a lightweight VM behind the scenes, and the container runs inside that VM.

Each Kata-launched container (or pod) thus has its own kernel and operating environment, isolated from the host at the hardware virtualization level. This extra layer (VM boundary) dramatically strengthens security. If a Kata container is compromised, the attacker is still trapped within a VM that has its own kernel, memory, and networking stack separate from the host's. Moreover, Kata Containers uses modern CPU virtualization extensions (Intel VT-x/VT-d or AMD-V) to enforce this isolation with minimal performance overhead.

From a theoretical standpoint, Kata Containers addresses a key weakness of traditional containerization by reintroducing the VM boundary without sacrificing too much of the agility.

The Kata isolation model is closer to running a microVM per container. As a result, workloads are better protected.

This level of isolation is especially beneficial in cloud environments or multitenant Kubernetes clusters, where untrusted code might be running. The trade-off is a slight increase in resource usage because each container has the overhead of a tiny VM; however, Kata Containers is optimized to keep this overhead as low as possible with the use of paravirtualized devices and lightweight hypervisors to maintain performance comparable to standard containers.

Architecture

At the heart of Kata Containers is an OCI-compliant container runtime (`kata-runtime`) that interfaces with container engines (like Docker or containerd) on the host side. When a container is started under Kata, `kata-runtime` launches a hypervisor (e.g., Qemu, Cloud Hypervisor, Firecracker) to create a new VM for the container. This VM is extremely

Lead Image © puttipongsong, 123RF.com

minimal because it uses a special lightweight guest kernel (often a pared-down Linux kernel optimized for containers) and a tiny root filesystem (or `initramfs`) that has just enough to run the container workload.

Inside the VM is a `kata-agent` process that is responsible for receiving container lifecycle commands from the host and orchestrating the setup inside the VM. Communication between the host's Kata runtime and the guest `kata-agent` happens over a secure channel (typically with virtio-vsock or the like, which provides a socket connection into the VM without needing a network connection).

It's worth noting that Kata Containers was born from the merger of the Clear Containers (Intel) and runV (Hyper.sh) projects. As a result, it inherited support for multiple CPU architectures (x86_64, AArch64/ARM64, IBM System p series, etc.) and can work with several hypervisors. By default on x86, Qemu with KVM is commonly used, but Kata can also use Firecracker (an AWS-derived microVM hypervisor) or Cloud Hypervisor, among others.

The choice of hypervisor can be configured in Kata's settings. Kata's design ensures that container orchestration tools see a standard Open Container Initiative (OCI) runtime. Whether you run `docker run` or deploy a pod in Kubernetes, you simply specify the Kata runtime, and Kata takes care of the complex task of launching a VM behind the scenes and integrating it into the container ecosystem. This compatibility with OCI and Kubernetes container runtime interface (CRI) means you can slot Kata into existing container environments with minimal changes, an important feature for practicality.

Preparing the Environment

Before installing Kata Containers on an Ubuntu server, you need to address a few prerequisites and considerations. First and foremost, hardware virtualization support must be enabled on the host. Because Kata uses KVM under the hood, your CPU needs to support virtualization extensions (Intel VT-x/VT-d or AMD-V, which should be enabled in BIOS/UEFI). On a Linux system, you can verify whether the host CPU supports virtualization by checking `/proc/cpuinfo`:

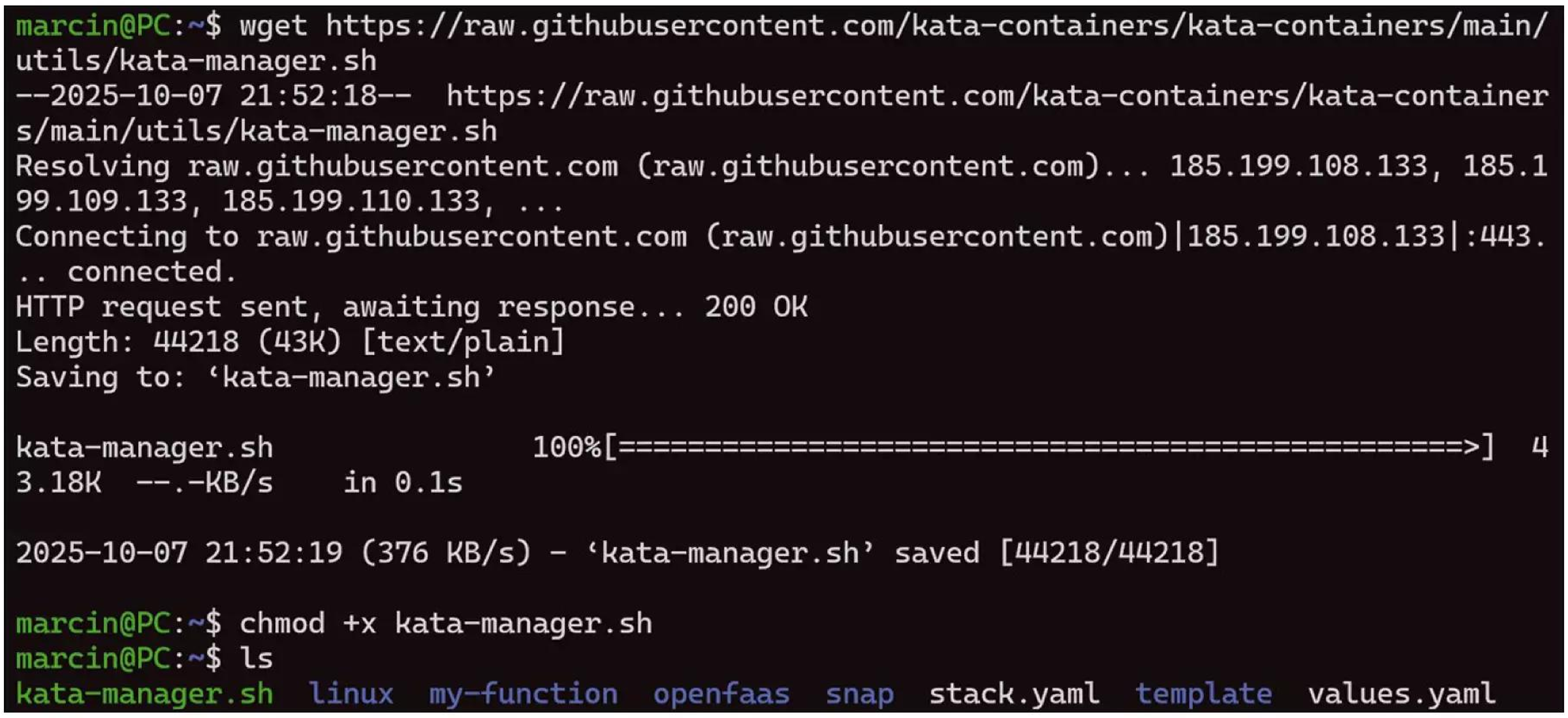
```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

If this command returns a number greater than *0*, your CPU has the necessary virtualization extensions. If it returns *0*, either the CPU doesn't support virtualization or it is disabled in the BIOS, in which case you must enable it before proceeding. Also, ensure the KVM kernel modules are loaded (`kvm_intel` or `kvm_amd`, depending on your processor). On Ubuntu, installing the *qemu-kvm* package usually sets this up; you can confirm by running

```
lsmod | grep kvm
```

If the host itself is a VM (e.g., you're setting up Kata inside a cloud VM or a VM on VMware/VirtualBox), then nested virtualization has to be enabled. Nested virtualization allows a VM to act as a hypervisor for other VMs. Many cloud providers support nested virtualization on certain instance types (AWS EC2 has specific instance families that allow nested KVM; Azure supports it on certain VMs).

Another prerequisite is having a container engine and runtime environment ready. In this example, I use Docker as the container engine (Docker includes its own container runtime by default, but I'll configure it to use Kata as an alternative runtime). Ensure that Docker is installed on your system. Also, have root or sudo privileges available, because installing Kata involves kernel-level components and daemon configuration changes.



```
marcin@PC:~$ wget https://raw.githubusercontent.com/kata-containers/kata-containers/main/
utils/kata-manager.sh
--2025-10-07 21:52:18--  https://raw.githubusercontent.com/kata-containers/kata-container
s/main/utils/kata-manager.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.1
99.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 44218 (43K) [text/plain]
Saving to: ‘kata-manager.sh’

kata-manager.sh               100%[=================================================>]  4
3.18K  --.-KB/s    in 0.1s

2025-10-07 21:52:19 (376 KB/s) - ‘kata-manager.sh’ saved [44218/44218]

marcin@PC:~$ chmod +x kata-manager.sh
marcin@PC:~$ ls
kata-manager.sh  linux  my-function  openfaas  snap  stack.yaml  template  values.yaml
```

Figure 1: Downloading the `kata-manager.sh` script from GitHub.

Installing Kata Containers

Kata Containers can be installed on Ubuntu in a few ways. One approach is to use distribution packages. In fact, Kata provides packages in an Open Build Service (OBS) repository for Ubuntu, which can be enabled and installed by Apt. I outline an automatic installation method that uses the Kata project's provided script, because this method ensures you get a properly configured latest version.

The Kata Containers community provides the `kata-manager.sh` script, which automates installation. On an Ubuntu 22.04 or 24.04 host, for example, you can retrieve this script from Kata's GitHub repository (Figure 1) with:

```
wget https://raw.githubusercontent.com/➘
  kata-containers/kata-containers/main/➘
  utils/kata-manager.sh
chmod +x kata-manager.sh
```

This script can install Kata with various options. By default it may set up Kata with containerd. Because I plan to use Docker, I can instruct the script accordingly. For instance, running

```
./kata-manager.sh -D </C>
```

(Figure 2) will install Kata and configure it for Docker usage, where the `-D` flag stands for Docker integration. If you want to install Kata for use with containerd (or Kubernetes) only, you could run it without `-D` or use `-o` to install only Kata without any specific container runtime integration.

With Kata now installed on the system, the next step is to configure Docker to use Kata as an available runtime. Docker uses an OCI runtime (runC by default) to launch containers. You can register Kata as an additional OCI runtime so that Docker can invoke it when needed.

Docker Daemon

Configuring Docker's daemon to add Kata as a runtime is done by editing Docker's daemon configuration (usually `/etc/docker/daemon.json`). If that file is not present, create it. To add Kata, include a `runtimes` section:

```
{
  "runtimes": {
    "kata": {
      "path": "/usr/bin/kata-runtime"
    }
  }
}
```

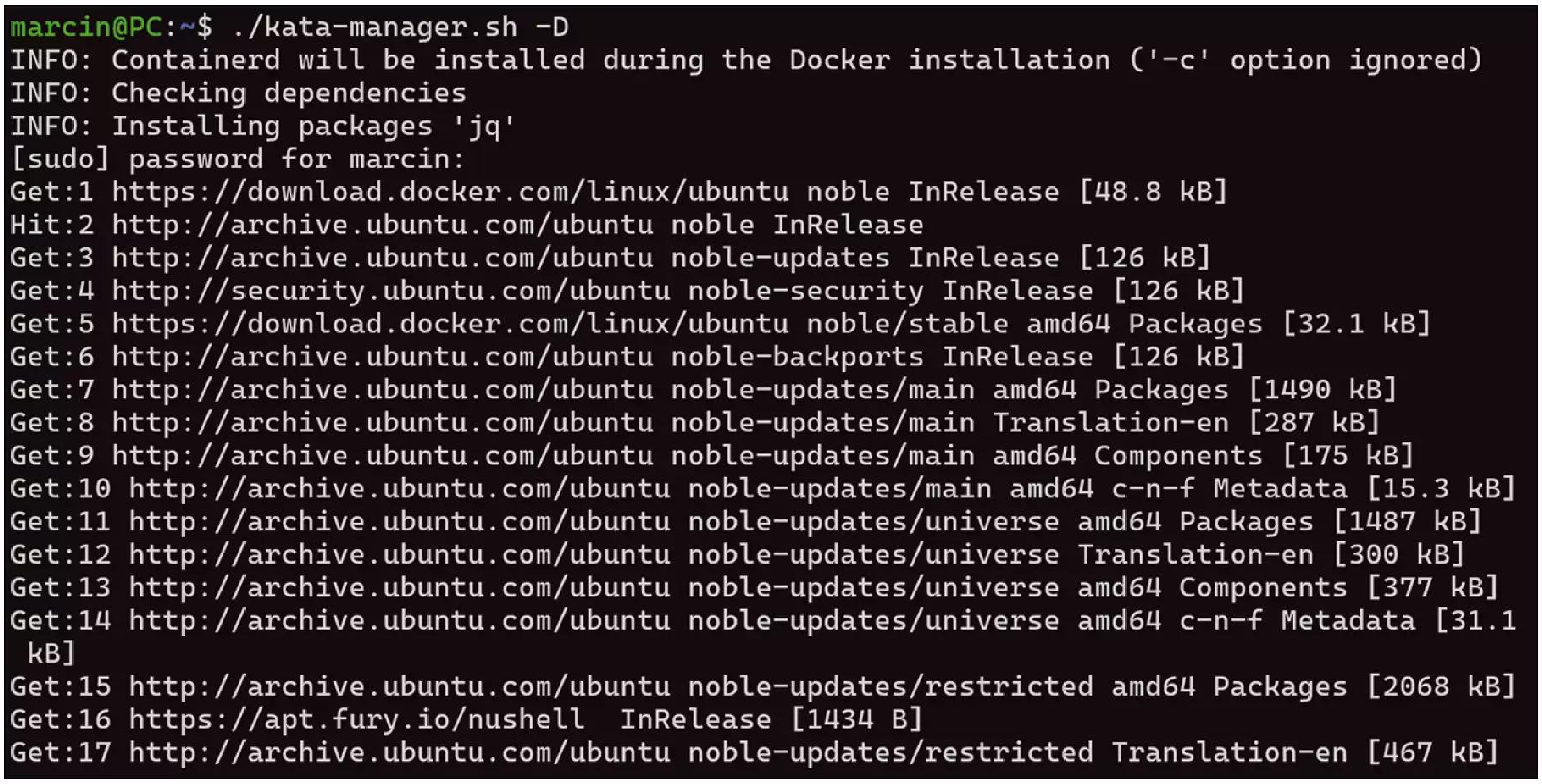
This snippet tells Docker that a runtime called `kata` is available and the command to invoke it is `/usr/bin/kata-runtime`. Once added, you need to restart the Docker daemon. Docker should now recognize the new runtime.

If you want all containers by default to use Kata, you can also set `default-runtime: "kata"` in the same JSON file. However, making Kata the default for all containers isn't usually desired unless you're in an environment in which every single container must be in a VM (e.g., a locked-down multitenant environment). Often, you'll want to use Kata selectively for certain containers that need the extra isolation and use the normal runC for others. In that case, do not set it as default; you'll specify the runtime per container at runtime.

Testing the Setup

Try running a container with Kata and see if it indeed uses a different kernel (confirming it's inside a VM):

```
# Run a test container with default ➘
  runtime (runc)
docker run --rm alpine uname -r
# Then run a test container with ➘
  Kata runtime
docker run ➘
  --rm --runtime=kata alpine uname -r
```



```
marcin@PC:~$ ./kata-manager.sh -D
INFO: Containerd will be installed during the Docker installation ('-c' option ignored)
INFO: Checking dependencies
INFO: Installing packages 'jq'
[sudo] password for marcin:
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [32.1 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1490 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [287 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.3 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1487 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [300 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.1
 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2068 kB]
Get:16 https://apt.fury.io/nushell  InRelease [1434 B]
Get:17 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [467 kB]
```

Figure 2: Installing Kata and configuring for Docker.

Now that Docker is integrated with Kata, you can choose which containers to run under Kata on a case-by-case basis. For example, if you have a particularly sensitive application (say, a third-party service you don't fully trust or a workload handling untrusted data), you might run it with Kata to give it that extra isolation. Meanwhile, your other containers can continue with the normal runtime to keep overhead low. In this way, you don't need a separate VM environment for those special cases; your container platform can handle both types of workloads easily.

Conclusion

Kata Containers provide a sophisticated yet practical approach to enhance container security by blending virtualization with containerization. By providing stronger workload isolation, Kata Containers adds an important security layer when managing containerized environments, especially in multitenant or sensitive deployments. At the same time, Kata preserves compatibility with existing tools and workflows (Docker, Kubernetes, etc.), making it a compelling option when you need to harden your container infrastructure without sacrificing too much in performance or flexibility.

In practice, adopting Kata Containers can be as simple as using a different runtime for specific containers, delivering substantial security benefits: Each container or pod becomes a true sandbox at the hardware level. As with any technology, it's important to keep Kata up to date and monitor its developments (the community is active under the OpenInfra Foundation). Kata Containers has evolved significantly since its introduction in 2017, and it continues to improve in performance and security. ■

Info

[1] Official documentation: [https://katacontainers.io/]

The Author

Marcin is a Microsoft MVP, Microsoft Certified Trainer, and conference speaker. He works as a DevOps architect and has extensive experience in Azure technologies. Visit his blog at [https://marcingastol.com/].

Shop the Shop shop.linuxnewmedia.com/shop

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com/shop

GET IT NOW!

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS

ADMIN Network & Security
Non-Relational Databases
Storage
Monitoring
Data Obfuscation
5 Network Admin Distros
Lightweight Kubernetes

Windows Administrator Protection replaces UAC

The Weakest Link

Administrator Protection replaces the deprecated User Account Control with state-of-the-art protection that uses the principle of least privilege and just-in-time privilege assignment. By Thomas Joos

Microsoft unveiled its new Secure Future Initiative (SFI) for Windows at Microsoft Ignite. The focus is on Administrator Protection, which is set to replace the deprecated User Account Control. Current statistics from the Microsoft Digital Defense Report 2024 [1] show an alarming 39,000 incidents of admin rights abuse every day, making better protection for local accounts a top priority.

Microsoft introduced User Account Control (UAC) in Windows Vista. Although it requires users to confirm actions with admin authorization, the protection turned out to be incomplete. Microsoft is now looking to improve protection for Windows 11 workstations and replace UAC with Administrator Protection [2]. In this article, I show you what this new feature is all about and present several arguments to explain why switching to a state-of-the-art protection setup makes sense.

At its core, Administrator Protection is based on the principle of least privilege. This new approach replaces permanent admin accounts that have excessive privileges with a system-managed admin account (the Super Administrator account) that only grants privileges dynamically when they are needed; otherwise, the account is completely protected.

UAC Weaknesses

User Account Control in Windows 11 distinguishes between standard and administrator accounts. When a UAC prompt appears, an access token is created granting the administrative privileges required for the requested action. The biggest weakness of this approach is that the extended token is permanently assigned to the process until it is terminated, which opens up opportunities for attackers to carry out token-spoofing attacks that exploit the extended rights of an existing process to carry out malicious actions.

The UAC bypass issue is another major headache, wherein attackers exploit vulnerabilities in trusted Microsoft signatures or system processes to bypass UAC queries and escalate their privileges. Processes such as `eventvwr.exe` or manipulated tasks in the task scheduler are particularly popular attack vectors. Another problem is that the UAC pop-up display is not isolated but appears in the user context, which allows attackers to gain administrative control in the process, called by attack vectors such as DLL injection. Last but not least, UAC's user friendliness is also a problem. Too many confirmation requests mean that users likely simply confirm UAC messages without reading them.

Convenient and Secure

Administrator Protection will initially be integrated into Windows Hello and Hello for Business. Users can confirm messages either with their fingerprint scanner or by looking at the camera. Definitely more secure than UAC, it is also convenient. Entering a PIN is also possible, just as when you log into Windows.

Authentication is handled by encryption keys stored in the Trusted Platform Module (TPM), removing the need for passwords. Hello for Business, the solution for corporate networks, integrates a public key infrastructure (PKI) to meet security-critical requirements such as multifactor authentication. The extended version of Windows Hello also supports hybrid scenarios with Entra ID/EntraPass and Active Directory, enabling seamless single sign-on across networks and cloud services. Unlike Windows Hello, which is limited to individual devices, Hello for Business scales across multiple devices, and it relies on key pairs instead of PINs, providing significantly stronger authentication for businesses.

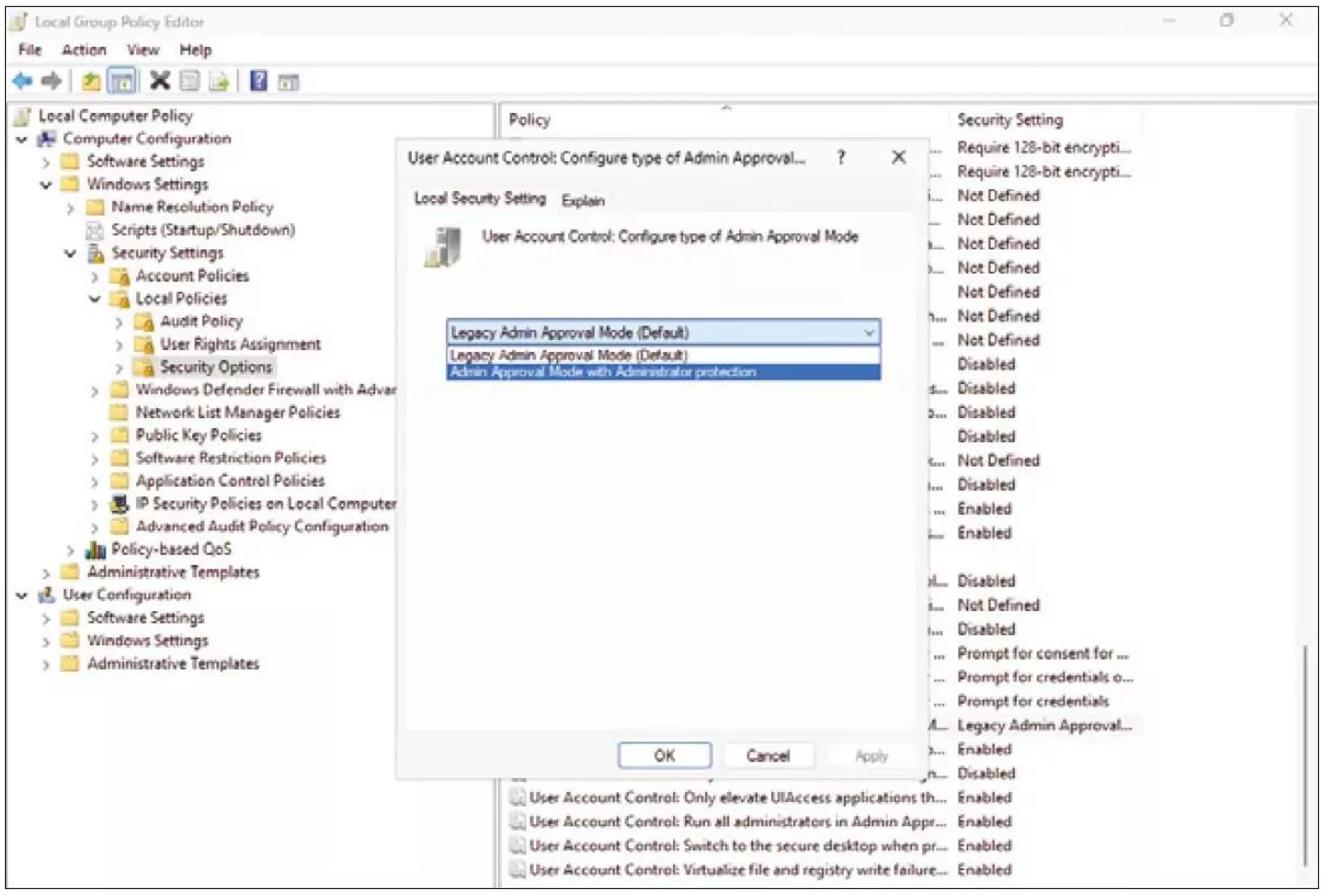
Lead Image © Romolo Tavani, 123RF.com

Windows Hello is also more secure than entering a password because it is based on asymmetric cryptography. Instead of a password, you use a private key; the key is stored in the device's TPM and never leaves it. In the next step, this key is combined with a public key pair to authenticate. Because no passwords are transmitted or stored, phishing and brute force attacks are ineffective. Moreover, biometric features such as facial recognition or fingerprinting enable unique identification that cannot be easily copied or guessed.

Granular Admin Rights

Unlike UAC, after enabling Administrator Protection, the user is not given admin rights for the entire session of the respective application after confirmation, but only for a single task. Administrator Protection adheres to least privilege and just-in-time principles. Necessary authorizations are only retained for as long as they are needed. Renewal is not automatic. Windows 11 users always work without admin rights. If an administrative task requires elevated rights, Administrator Protection assigns an admin token that applies to the task at hand only, and it expires after the action completes. Therefore, every administrative activity requires a new admin token, including installing software, changing system settings (e.g., the time or the registry), and accessing sensitive data.

Administrator Protection uses specially protected accounts that are completely separate from the system. This principle means that malware has no chance of hijacking the accounts, because they cannot be compromised in the conventional user context. Microsoft is convinced that the new technology is far more difficult to work around than UAC. Even if an attacker manages to compromise the user with Administrator Protection once, they cannot hijack the entire system. The new security features in Windows 11 prevent automatic and direct access to the kernel or other critical system resources.


Local Group Policy Editor
File Action View Help
Local Computer Policy
Computer Configuration
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Security Settings
Account Policies
Local Policies
Audit Policy
User Rights Assignment
Security Options
Windows Defender Firewall with Advar
Network List Manager Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
IP Security Policies on Local Computer
Advanced Audit Policy Configuration
Policy-based QoS
Administrative Templates
User Configuration
Software Settings
Windows Settings
Administrative Templates
Policy
Security Setting
OK
Cancel
Apply

User Account Control: Configure type of Admin Approval...

Local Security Setting | Explain

User Account Control: Configure type of Admin Approval Mode

Legacy Admin Approval Mode (Default)

- Legacy Admin Approval Mode (Default)
- Admin Approval Mode with Administrator protection

User Account Control: Only elevate UIAccess applications th... Enabled
User Account Control: Run all administrators in Admin Appr... Enabled
User Account Control: Switch to the secure desktop when pr... Enabled
User Account Control: Virtualize file and registry write failure... Enabled

Figure 1: Setting Administrator Protection as the admin approval method.

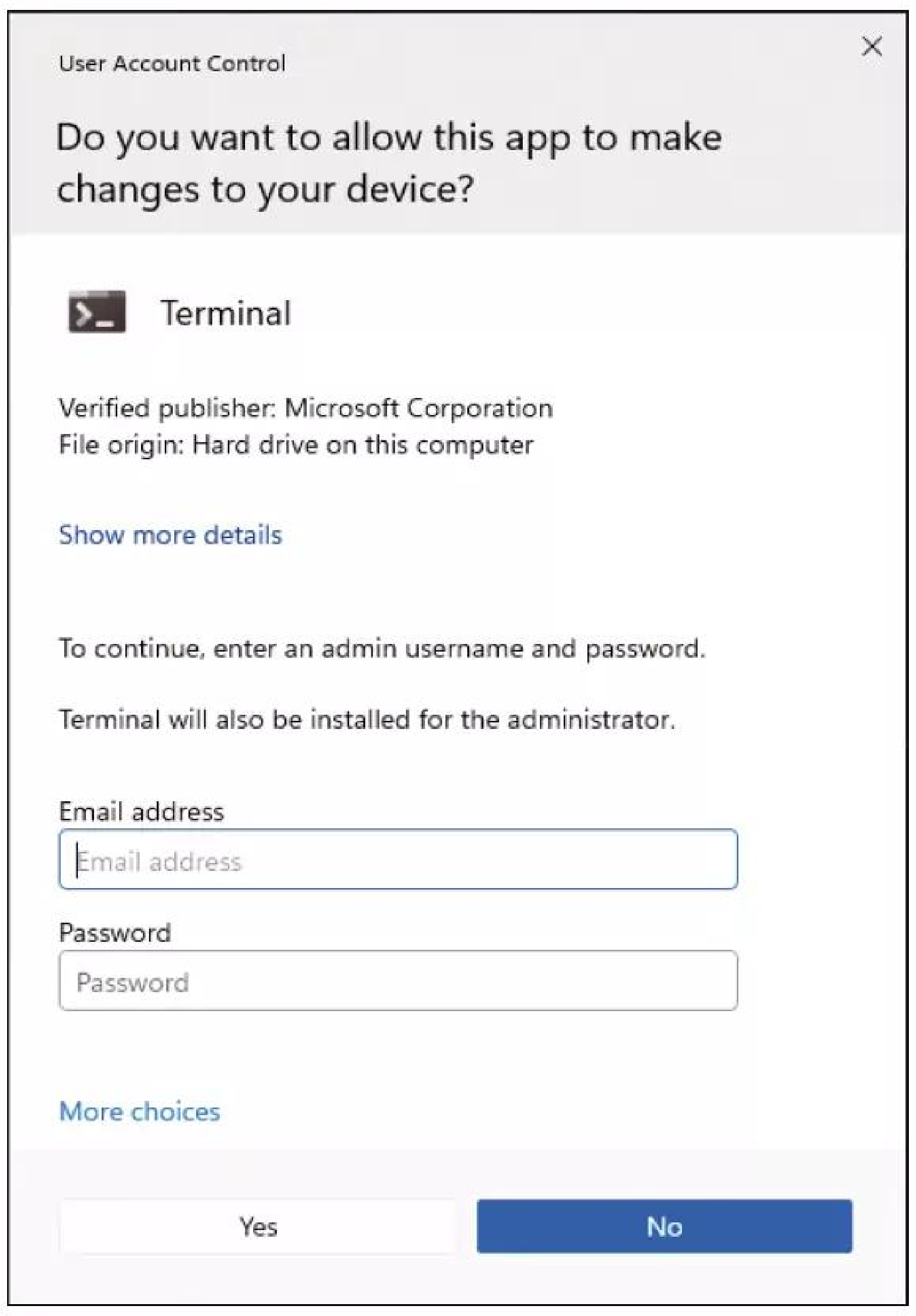
Enabling Administrator Protection

A preview of the new technology for protecting admin accounts is available in the latest Canary previews of Windows 11 from version 27718 and newer. Besides enabling from the *Account protection* selection in the Windows Security app, this feature can be enabled by group or local policies. The settings are configured in *Computer Configuration | Windows Settings | Security Settings | Local Policies | Security Options* in the *User Account Control: Configure type of Admin Approval Mode* policy setting (Figure 1).

The default setting is *Legacy Admin Approval Mode (Default)*. To use the new mode, you need to enable *AdminApproval Mode with Administrator protection* and then restart so Windows 11 can use the feature. After activation, a new folder named `C:\Users\ADMIN_<username>` appears. Account protection uses this folder to handle permissions. You can test for activation by starting the Windows Terminal with admin rights. If the new protection is activated, Windows 11 will pop up a window to confirm the necessary permissions (Figure 2).

Windows Strategy Versus Linux Root

Conceptually, the new Administrator Protection in Windows 11 is based on security principles that have been established in the Linux environment for years, but the implementation is different. On Linux systems, the root account is the central instance for administrative tasks, with access to this account remaining clearly limited. Actions that require root privileges are performed by mechanisms such as `sudo` that allow users to execute


User Account Control
Terminal
Show more details
More choices
Yes
No

Do you want to allow this app to make changes to your device?

Verified publisher: Microsoft Corporation
File origin: Hard drive on this computer

To continue, enter an admin username and password.

Terminal will also be installed for the administrator.

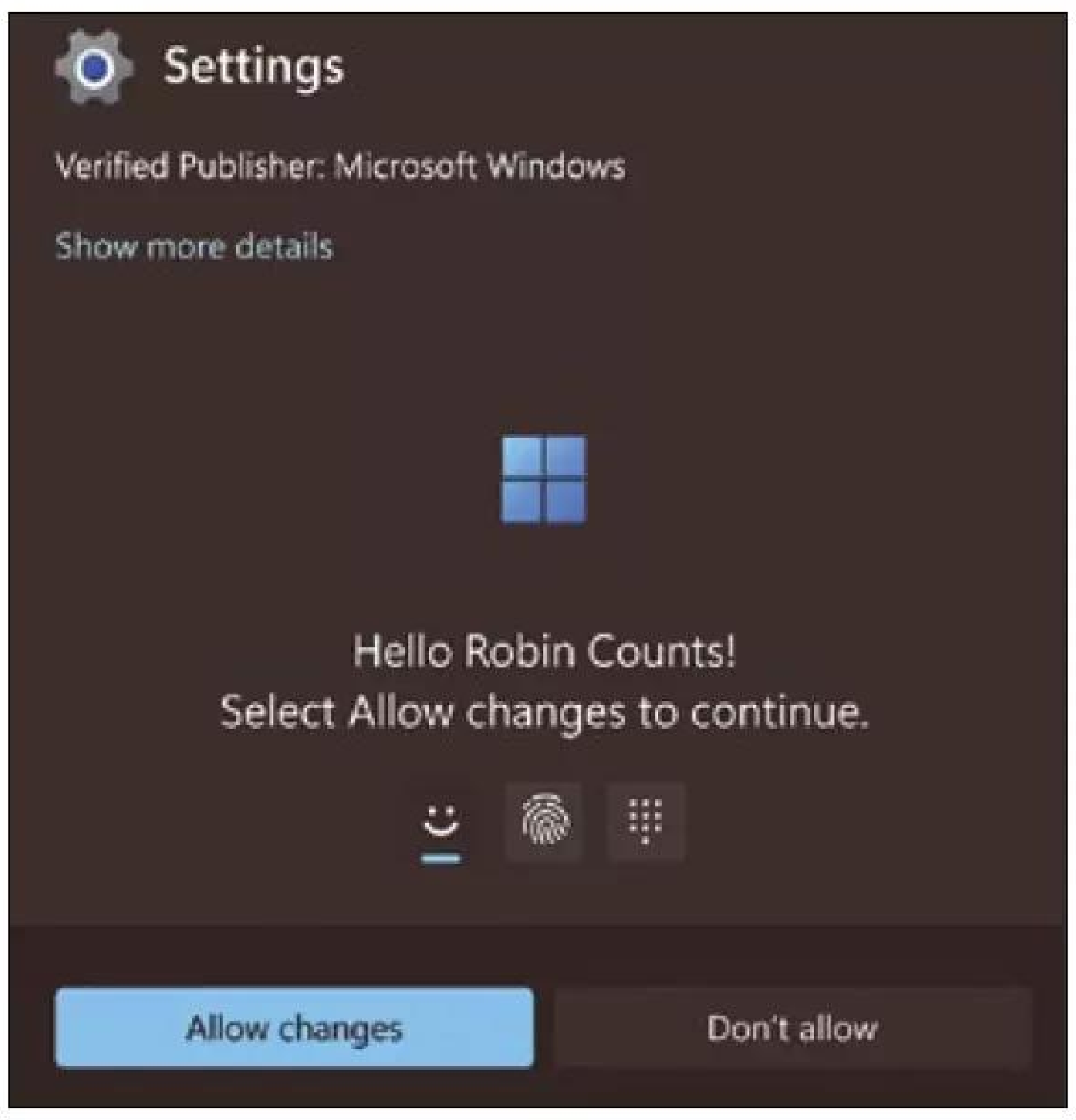
Email address: Email address
Password: Password

Figure 2: Windows 11 will have the new Administrator Protection in the future and do away with the less secure UAC.

privileged commands temporarily without permanently possessing root privileges. Administrator Protection reduces the risk of abuse in a similar way by granting privileges on a just-in-time basis. Users are only given a temporary token for individual tasks, and the token expires on completion. Although Linux also consistently follows the principle of least privileges, Windows 11 extends it by integrating biometric methods courtesy of Windows Hello (Figure 3). Microsoft combines the security advantages of role-based access control with modern authentication methods such as fingerprint or facial recognition. One key difference lies in the management of privileged accounts: Administrator Protection uses system-managed accounts that operate in isolation from the user context, reducing the attack surface for malware. In contrast, the root account in Linux is directly accessible and requires exacting security measures by the administrator.

Both approaches aim to minimize the attack surface, with Windows 11 placing a stronger focus on user-friendliness through the integration of TPM-based keys and biometric multifactor authentication. The Administrator Protection approach is more secure than the traditional root account in Linux in certain respects, especially in scenarios with less experienced users or in companies seeking greater standardization and automation.

Windows 11 only grants administrative rights temporarily and only for individual tasks. These rights expire automatically on completion of the task, which minimizes the risk of a compromised process or user with administrative authorizations gaining permanent access to critical systems. Linux offers a similar function in the form of `sudo`, but in practice, many administrators disable logging of `sudo` commands or use root sessions, although doing so partially circumvents the intended security feature. Biometric methods such as facial recognition or fingerprinting make misuse by third parties more difficult. Linux can achieve comparable security mechanisms through external configuration that involves manual overhead and is less standardized.



Verified Publisher: Microsoft Windows

Hello Robin Counts!
Select Allow changes to continue.

Figure 3: The new Administrator Protection supports Windows Hello for greater security and convenience.

Revised Malware and Kernel Protection

In addition to the new account protection, Microsoft is revising other security features in Windows. As a consequence of the CrowdStrike incident in the summer of 2024, in which millions of PCs and servers were rendered inoperable by a faulty update to CrowdStrike's security solution, Microsoft is now significantly restricting access to the kernel – for antivirus software, as well. In future, Windows will no longer run in kernel mode but in user mode, although it will still have access to system resources. Administrator Protection also benefits, because it ensures superior protection against attacks on the kernel.

Windows 11 isolates kernel mode from user mode to isolate critical system functions and applications. Kernel mode provides unrestricted access to the hardware and all system resources. In contrast, user mode is restricted to applications and services that are not allowed to interact directly with the operating system. An attacker gaining access to kernel mode can manipulate the operating system in depth, including installing rootkits, bypassing security mechanisms, and monitoring all system traffic.

If kernel mode is compromised, the attacker gets complete control over the system with all the serious security implications this entails. Therefore, Windows 11 has protected kernel mode for some time with proven mechanisms such as virtualization-based security (VBS), hypervisor-protected code integrity (HVCI), and Secure Boot. These features reliably prevent unauthorized access and manipulation. The new restrictions relating to the kernel improve this protection against attacks. Secure Future Initiative features also ensure that faulty updates to security products can no longer take down the entire operating system.

Conclusions

In Windows 11, Administrator Protection replaces the deprecated UAC with a state-of-the-art and more secure approach that uses the principle of least privilege and just-in-time privilege assignment. Administrative privileges are only assigned temporarily and on a task-specific basis, significantly reducing the risk of misuse with permanent admin rights assignments. Administrator Protection also integrates biometric authentication on Windows Hello, including fingerprint, facial recognition, and TPM-secured encryption. Unlike UAC, Administrator Protection enables complete separation of user and admin accounts, making it even more difficult for malware to gain privileged access.

Info

[1] Microsoft Digital Defense Report 2024: [https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024]

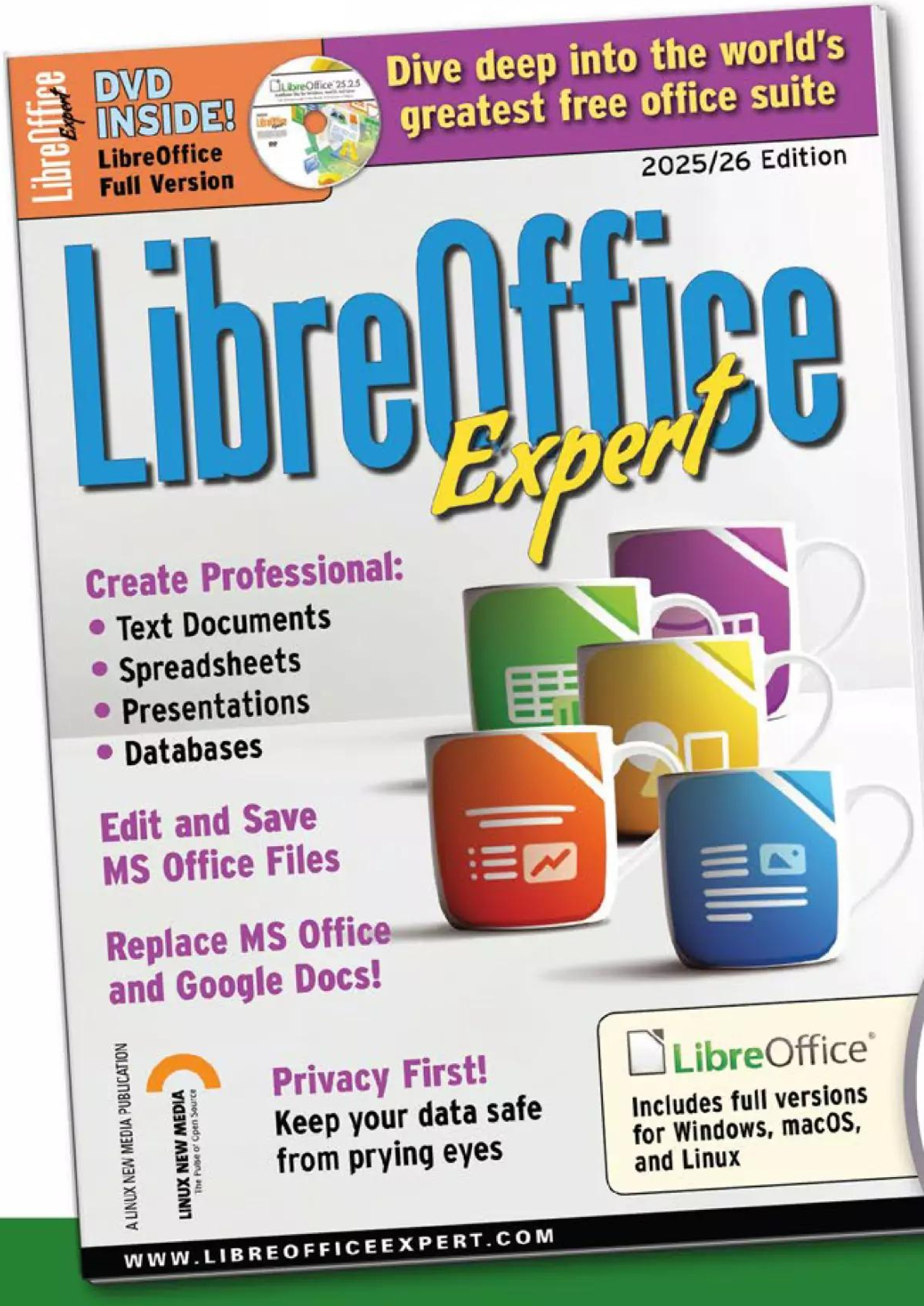
[2] Administrator Protection: [https://techcommunity.microsoft.com/blog/windows-itpro-blog/administrator-protection-on-windows-11/4303482]

The Author

Thomas Joos is a freelance IT consultant and has been working in IT for more than 20 years. In addition, he writes hands-on books and papers on Windows and other Microsoft topics. Online you can meet him on [http://thomasjoos.spaces.live.com].

Shop the Shop
shop.linuxnewmedia.com/shop

Become a LibreOffice Expert



LibreOffice Expert
DVD INSIDE!
LibreOffice Full Version
Dive deep into the world's greatest free office suite
2025/26 Edition

LibreOffice Expert

Create Professional:
- Text Documents
- Spreadsheets
- Presentations
- Databases

Edit and Save MS Office Files

Replace MS Office and Google Docs!

Privacy First!
Keep your data safe from prying eyes

LibreOffice
Includes full versions for Windows, macOS, and Linux

A LINUX NEW MEDIA PUBLICATION
LINUX NEW MEDIA
The Pulse of Open Source

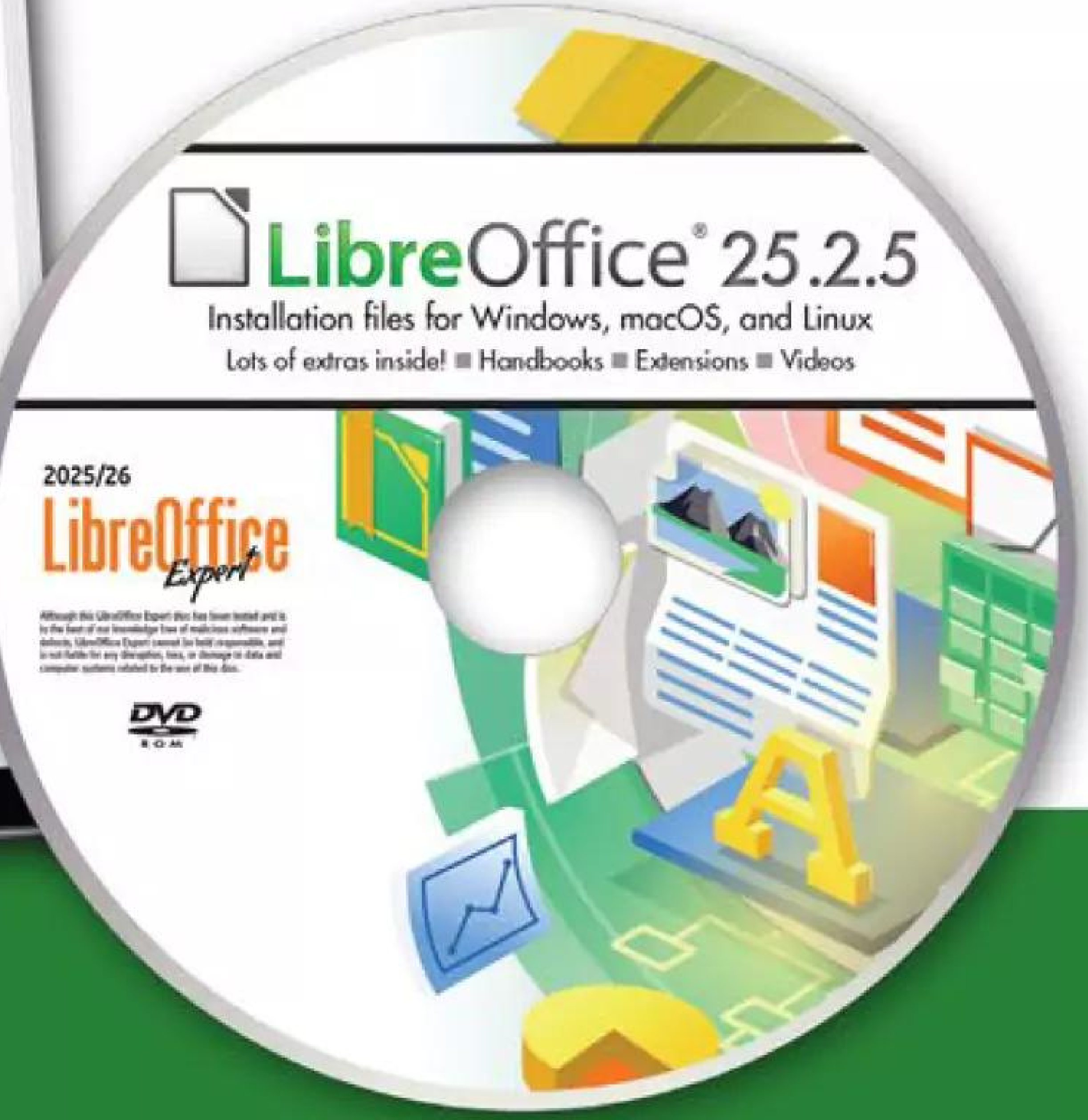
WWW.LIBREOFFICEEXPERT.COM

Explore the FREE office suite used by busy professionals around the world!

Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!



LibreOffice 25.2.5
Installation files for Windows, macOS, and Linux
Lots of extras inside! ■ Handbooks ■ Extensions ■ Videos
2025/26
LibreOffice Expert
DVD ROM

Order online:
shop.linuxnewmedia.com/shop

For Windows, macOS, and Linux users!

Photo by Declan Sun on Unsplash

Explore the Tetragon security suite for Kubernetes

Inside the Polygon

Tetragon is a comprehensive security suite for Kubernetes based on eBPF and focused on monitoring and proactive threat detection. The tool also offers tracing for popular programming languages. By Martin Loschwitz

Attack scenarios are becoming increasingly complex, as are the infrastructures on which organizations root their IT. Kubernetes, with its many layers for storage and networking, along with the microarchitecture applications running on it, is a prime example of ever-increasing complexity. Unsurprisingly, security is one of the key issues for IT managers.

Clearly, new tools are needed. In recent years, new solutions have emerged that let you detect threats and attacks more easily in scalable environments. A previous *ADMIN* article on Falco [1] [2] is one example, and Tetragon [3], which I discuss in this issue, is a direct competitor.

Much like Falco, Tetragon promises proactive monitoring of containerized setups in Kubernetes, but that is not the only similarity: Like its competitor, Tetragon relies heavily on the extended Berkeley Packet Filter (eBPF) virtual kernel network engine to acquire and analyze ongoing packet flows. This relationship makes sense, because Tetragon was created by the developers of Cilium, one of the best-known implementations of software-defined networking (SDN) for Kubernetes.

Tetragon is designed to handle a wider range of tasks than Falco. In addition to monitoring programs and data streams at runtime, it offers comprehensive options for monitoring, alerting, and trending. It also has a tracing function that can be deployed at the program level. In this article, I present Tetragon in full detail and discuss the tool's strengths and weaknesses.

Complex Architecture

An architecture diagram (Figure 1) also helps in understanding the solution that Tetragon offers. You can quickly see the parallels with Falco, but also significant differences.

Both tools rely on eBPF, which is an in-kernel virtual machine (VM) that is not designed to run generic virtual machines like KVM. Instead, eBPF executes small programs dedicated to network data, linking them directly to the respective data streams. eBPF's performance is particularly impressive. Its architecture enables it to divert or manipulate network traffic in near real time.

Tetragon enters the scene here by storing eBPF programs in the Linux kernel. The programs, in turn, use hook points defined in eBPF (i.e., specific events on which eBPF focuses when routing traffic). One of three hook points can be chosen: `tracepoint` for tracking packet streams, `kprobe` for events related to a Linux kernel system call, and `uprobe` for targeting program calls in userspace.

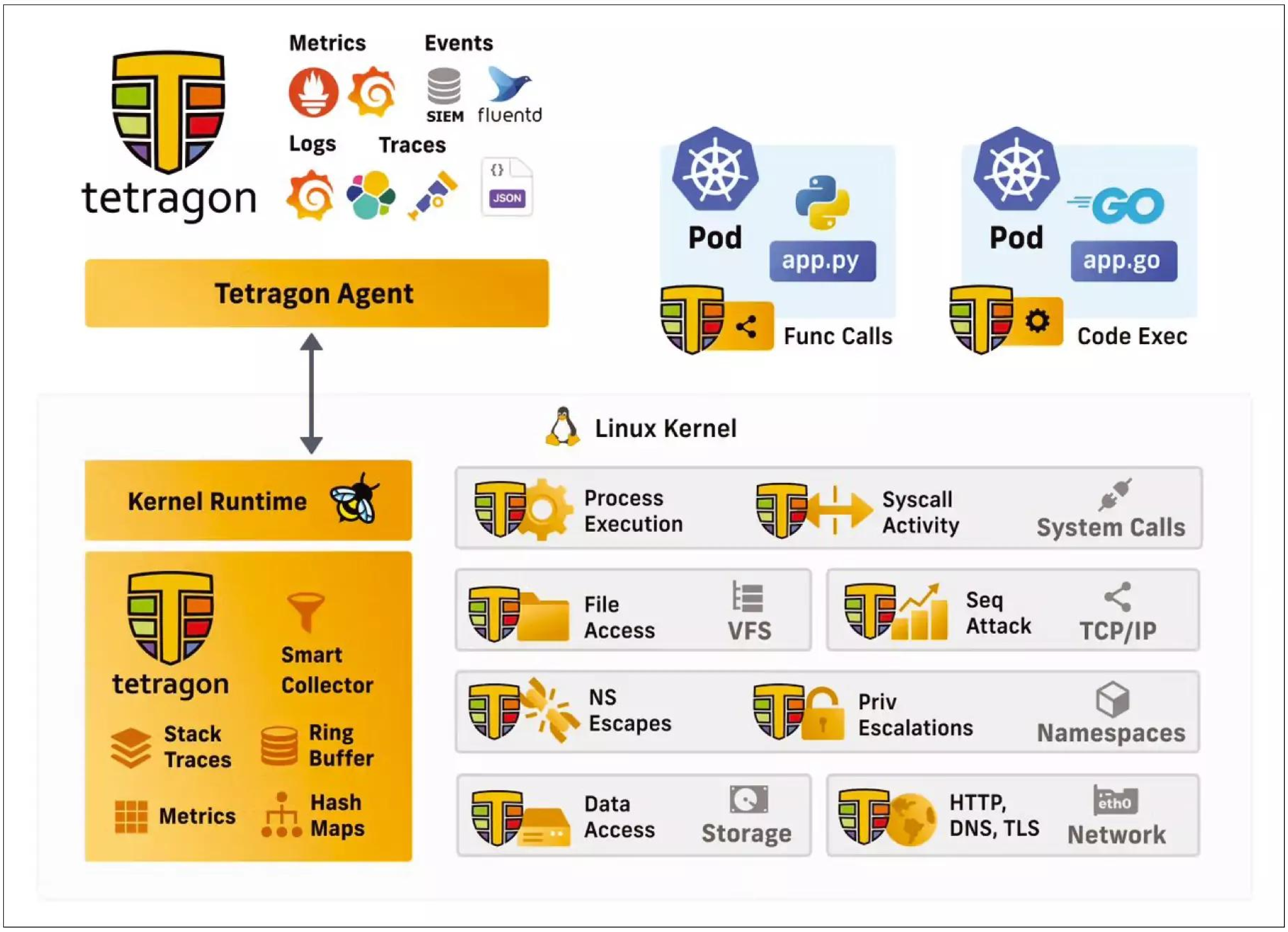

tetragon
Metrics
Events
SIEM
fluentd
Logs
Traces
JSON
Tetragon Agent
Pod
app.py
Func Calls
Pod
GO
app.go
Code Exec
Linux Kernel
Kernel Runtime
tetragon
Smart Collector
Stack Traces
Ring Buffer
Metrics
Hash Maps
Process Execution
Syscall Activity
System Calls
File Access
VFS
Seq Attack
TCP/IP
NS Escapes
Priv Escalations
Namespaces
Data Access
Storage
HTTP, DNS, TLS
eth0
Network

Figure 1: Tetragon relies on eBPF to hook into the Linux kernel and implements functions for monitoring and active intervention. An additional feature is the comprehensive export options for the acquired data. © Isovalent

The `kprobes` hook points attach themselves to calls such as `system` or `execve` to capture the matching events. Restricting the focus with `uprobes` is more difficult: Any function defined by any program in userspace can be targeted. eBPF also distinguishes between intercepting `kprobes` and `uprobes` calls. The `kprobes` hook point can be recorded by eBPF natively and directly in the Linux kernel. In contrast, `uprobes` requires separate integration into the respective program; otherwise, data acquisition will not work. A `uprobes` hook point is the Tetragon feature set used for tracing data within applications. Together, `tracepoints`, `uprobes`, and `kprobes` are a powerful combination.

eBPF provides a generic programming interface so that programs can create their own functions and use them in the kernel. Tetragon takes a pragmatic approach and uses `matchArgs`, for example, to monitor specific values that are passed to programs and functions as arguments. With `matchPIDs`, active packet streams can be monitored in terms of the process IDs (PIDs) from which they originate. Additionally, `matchActions` allows selective filtering on the basis of specific events. These functions are the core of the tool.

Not Just Kubernetes

Although its developers provide Tetragon with a clear focus on Kubernetes, it can still be used effectively, but maybe less conveniently, outside of Kubernetes.

In the simplest Tetragon setup imaginable, you just install the Tetragon binary, which was written in Go, then launch it with root privileges and make sure a systemd unit keeps the tool running around the clock. The Tetragon daemon runs in the background. A dedicated command-line interface (CLI) command is also named `tetragon`. Even on a system without containers, the

Listing 1: eBPF Policy for Tetragon

```
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name:
"monitor-network-activity-outside-cluster-cidr-range"
spec:
  kprobes:
  - call: "tcp_connect"
    syscall: false
    args:
    - index: 0
      type: "sock"
    selectors:
    - matchArgs:
      - index: 0
        operator: "NotDAddr"
        values:
        - 127.0.0.1
        - ${PODCIDR}
        - ${SERVICECIDR}
```

```
tetragon getevents -o compact
```

command delivers a chronological overview of events relating to the start or termination of processes (Figure 2).

If you want to learn more about a system, you can check out the rulesets, which come in a clear-cut YAML format and are known as policies in Tetragon-speak. The Tetragon project itself provides a basic set of policies in its GitHub directory [4]. Among other things is a policy for intercepting and acquiring eBPF events. Additionally, a sample policy monitors tools that specifically target the SSH daemon (`sshd`), ensuring that Tetragon records an event when a process named `sshd` or `tcpserver` makes certain system calls, such as opening a TCP connection. Tetragon keeps records of the host that initiated the connection and the target. If a cybercriminal tries to connect to a system over SSH, Tetragon automatically detects this action and triggers an alert.

Listing 1 contains a short but meaningful example of a policy in Tetragon that is intended for use in Kubernetes, as you can clearly see from the reference to the network mask of a pod – a typical Kubernetes approach. Otherwise, the example is straightforward: If you apply the rule from the example, it tells you when outbound TCP connections to external hosts are opened.

The logic resides in the block starting `kprobes`; it defines an eBPF rule that takes effect when `tcp_connect` is used. This is not a Linux system call, but a standard C library function. Accordingly, the `syscall` parameter is set to `false`. The `selectors` then narrow down the selection. The `matchArgs` parameter introduces a section containing the restrictions.

The rule's consecutive number is specified by `index`. The `operator` specifies that the eBPF rule only applies if the target address (`DAddr`) is located on the active application's own pod network or on the network of the service address through which Kubernetes exposes the service to the outside world – and only if it is not 127.0.0.1 (i.e., `localhost`). These restrictive rules filter internal connections out of the results. In line with this, the eBPF program output exclusively comprises outgoing connections from the service to the outside world.

A `notSAddr` parameter can be used to implement a reverse scenario. The program then detects outgoing connections that do not originate from specific source addresses defined as trustworthy.

Powerful When Working Together

If you use Tetragon as a standalone binary, say, to monitor resources on a host, you are only using a fraction of the solution's capabilities. Tetragon is far more than an admin tool with its own daemon that detects and logs events on a system with the help of eBPF. It only unleashes its full potential when used in combination with Kubernetes and Helm.

Tetragon also comes with its own agent, which acts as a connection point for typical monitoring and tracing tools. The developers adhere to standards: If so desired, the Tetragon agent can communicate with Prometheus over a specially designed interface. Logfiles can be natively provided by Tetragon for Loki or Elasticsearch. An open observability interface is available for tracing, and the developers claim comprehensive connectivity to security information and event management (SIEM) environments and Fluentd.

A Question of Deployment

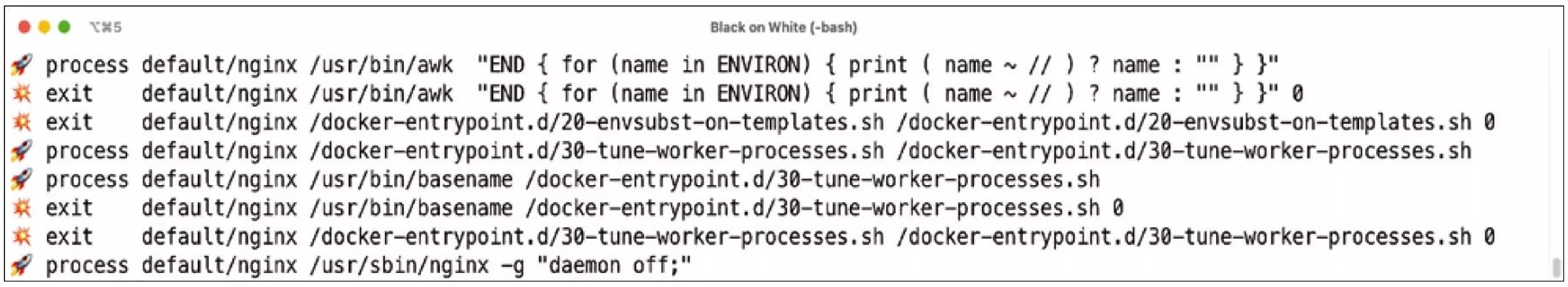
If you are wondering how Tetragon works in a larger context, the answer is that it depends on how you roll out and operate the tool in your environment, which, in turn, depends on the type of platform being used. Kubernetes is well known for providing standardized APIs for standard tasks. As long as you use Kubernetes and not some other container orchestrator (e.g., Nomad or Docker Swarm), you can take advantage of the comprehensive integration with Kubernetes that Tetragon itself provides. This integration is primarily based on Helm, the Kubernetes software manager.

The installation is quite simple. Three commands are all it takes to complete the basic Tetragon install process in a Kubernetes installation:

```
$ helm repo ➘
  add cilium https://helm.cilium.io
$ helm repo update
$ helm install tetragon cilium/tetragon ➘
  -n kube-system
```

Doing so adds several components to the environment. Most importantly, it includes a dedicated Kubernetes operator for Tetragon.

In Kubernetes-speak, an operator is a collection of custom resource definitions (CRDs). At its core, Kubernetes attempts to map all tasks handled at the data center to software and make them controllable by API. Because the Kubernetes API itself cannot cover



```
🚀 process default/nginx /usr/bin/awk  "END { for (name in ENVIRON) { print ( name ~ // ) ? name : "" } }"
💥 exit    default/nginx /usr/bin/awk  "END { for (name in ENVIRON) { print ( name ~ // ) ? name : "" } }" 0
💥 exit    default/nginx /docker-entrypoint.d/20-envsubst-on-templates.sh /docker-entrypoint.d/20-envsubst-on-templates.sh 0
🚀 process default/nginx /docker-entrypoint.d/30-tune-worker-processes.sh /docker-entrypoint.d/30-tune-worker-processes.sh
🚀 process default/nginx /usr/bin/basename /docker-entrypoint.d/30-tune-worker-processes.sh
💥 exit    default/nginx /usr/bin/basename /docker-entrypoint.d/30-tune-worker-processes.sh 0
💥 exit    default/nginx /docker-entrypoint.d/30-tune-worker-processes.sh /docker-entrypoint.d/30-tune-worker-processes.sh 0
🚀 process default/nginx /usr/sbin/nginx -g "daemon off;"
```

Figure 2: Tetragon can export events in JSON or by remote procedure call (gRPC), but command-line output offers the simplest view. Various program calls are documented here.

every conceivable scenario, however, it offers the option of storing CRDs. These definitions extend the Kubernetes API and teach it how to support additional functions.

Tetragon makes extensive use of this arrangement: Once the CRDs for Tetragon are stored in Kubernetes, Tetragon policies can be defined by the Kubernetes API, as can connections to various monitoring services. The Tetragon operator is therefore an integral part of the solution, but it is not the only component that comes with Helm.

The setup also includes a permanently active instance of Tetragon itself as a control daemon in the Kubernetes cluster. The daemon controls Tetragon across the entire Kubernetes installation in the form of a server. However, a server in Kubernetes requires a counterpart on the individual systems to field the instructions from the central server and convert them into specific configurations. To do this, the Tetragon program itself is deployed on the individual cluster nodes of the installation, but in agent mode.

Thanks to its comprehensive integration into Kubernetes by operators, Tetragon automatically takes care of factors such as redundancy and high availability. The Tetragon server instance, for example, is created as a Kubernetes `ReplicaSet`. If one instance of the service fails, Kubernetes automatically ensures that the service restarts elsewhere.

Superior Integration

As described earlier, unlike Falco, Tetragon follows a "Kubernetes-first" policy. More specifically, this means that Tetragon's integration with Kubernetes is significantly better than Falco's, delving deep into the inner sanctum of Kubernetes.

An example illustrates: Falco is based on the principle of monitoring running processes in Kubernetes, analyzing them in userland, and raising an alarm if necessary. Falco also connects to running processes by eBPF. However, Falco is more reserved when it comes to using the functions provided by eBPF. For example, eBPF has an option of not only capturing existing data streams and events in an eBPF program, but also influencing them. However, Falco restricts itself to raising alarms for certain events through the channels you define.

Tetragon takes a far more hands-on approach: Within the framework of a Tetragon policy, you have access to all the tools that eBPF offers, including the ability to intercept events actively on systems in an automated way.

Assume you want individual programs on your platform to communicate with certain external addresses only. This scenario is particularly common in environments with complex compliance requirements. In the past, it was common practice to examine only incoming connections and block them if necessary to prevent intruders from finding a gateway. Now, however, unauthorized traffic from the inside to the outside – indicating that a criminal has already succeeded in breaking in and is now actively stealing data or using the available resources for other illegal activities – is also considered a problem.

Falco makes it easy to identify this kind of undesirable data traffic. An alarm is triggered by connecting to tools such as Prometheus. As the next step, though, you need to check whether the identified traffic is really from an undesirable source and take action if appropriate. Tetragon offers more options: Of course, you can design a policy so that outgoing traffic is detected and triggers an alarm in the event of an incident, but with Tetragon you can also let eBPF block this traffic directly. You are still notified about the incident, but you do not have to intervene personally and immediately to avert a possible threat.

Another notable feature is that, because Tetragon relies on eBPF to analyze traffic and perform other actions directly, it is far faster than Falco. Falco derives event records from the Linux kernel's event API and then has to carry out the analysis step in the userland of the respective system. Tetragon, on the other hand, natively leverages the eBPF functions in the kernel.

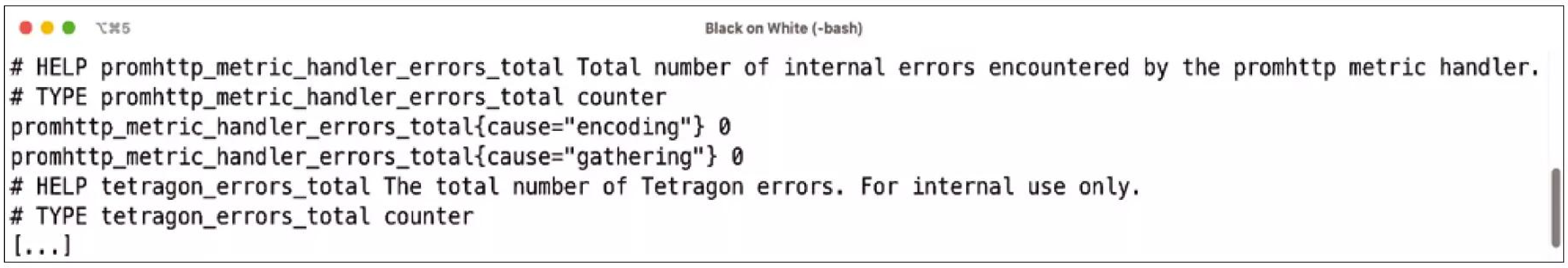
Estimates suggest that Falco requires eight to 10 percent of an environment's resources to perform its tasks; Tetragon has less than one-percent overhead. This lower overhead is an indication of the efficiency that eBPF now delivers and is clear proof that, despite its name, eBPF is more than just an advanced packet filter. In recent years it has instead evolved into a comprehensive tool for monitoring and enforcing compliance and security.

Performance is also a priority for the Tetragon developers in other areas. Because policies in Tetragon fundamentally understand Kubernetes and integrate with the orchestrator, they typically handle the integration of eBPF rules on the hosts with the Kubernetes API and the Kubernetes agents (Kubelets) on the target systems, which can cause short delays when starting the container at runtime (e.g., when a container needs to be started on a compute node, but the Kubelet is still busy with other tasks).

Tetragon provides a remedy with runtime hooks. When enabled, Tetragon directly connects to the runtime environments for containers on the compute nodes, usually Podman or Docker, and ensures that the containers implement the rules without delay. Under the hood, more services are not necessarily running in the background on the individual systems, but potential delays are beneficially eliminated.

Kubernetes First

The Kubernetes-first policy of its developers gives Tetragon further advantages in this regard. Because Tetragon is designed to work natively with Kubernetes, it understands its internal structure and docks onto it. Where alternative solutions such as Falco have to be rolled out by external tools such that they serve as sidecars for individual services in Kubernetes, Tetragon communicates natively with the Kubernetes API and interprets


⌥⌘5
Black on White (-bash)

```
# HELP promhttp_metric_handler_errors_total Total number of internal errors encountered by the promhttp metric handler.
# TYPE promhttp_metric_handler_errors_total counter
promhttp_metric_handler_errors_total{cause="encoding"} 0
promhttp_metric_handler_errors_total{cause="gathering"} 0
# HELP tetragon_errors_total The total number of Tetragon errors. For internal use only.
# TYPE tetragon_errors_total counter
[...]
```

Figure 3: Connecting to Tetragon over HTTP on port 2112 takes you to the metrics interface for Prometheus.

it accordingly, which is made clear when looking at examples of integration with other services.

Even though the previous section might have given the impression that event notification is merely a good-to-have extra in Tetragon, monitoring and alerting are high on the priority list for the solution's developers and is where the aforementioned integration with external tools comes into play.

In today's scalable environments, a combination of, say, Prometheus, its Alertmanager, and Grafana is considered an ideal solution for monitoring, alerting, and trending (MAT). Prometheus collects the metrics from the entire installation as a time series database and stores them [5]. On the basis of rules you define, Prometheus generates alerts when certain metrics values exist and passes them on to Alertmanager for delivery. Grafana also makes it easy to visualize the stored metrics.

For Kubernetes to collect metrics from a service in a meaningful way, though, the data must also be available in a suitable format. Both Falco and Tetragon take an autonomous approach in this case. Both tools have a metrics interface that queries Prometheus directly (Figure 3). However, major differences prevail when it comes to embedding Tetragon or Falco in Kubernetes: With Falco, it is a laborious task that takes up much of your valuable time.

Tetragon, on the other hand, integrates itself into the Kubernetes Prometheus stack, which can be quickly installed with the Prometheus operator, provided in turn by Helm. You just need a few entries in the Helm chart configuration for the respective cluster, and they are clearly described in the Tetragon documentation. Once these settings have been stored, though, a Prometheus instance rolled out by an operator in Kubernetes automatically detects the endpoints provided by Tetragon as the target for its queries. All that remains for you to do is create suitable dashboards in Grafana; and, again, suitable templates can be found in Grafana Marketplace [6] (Figure 4).

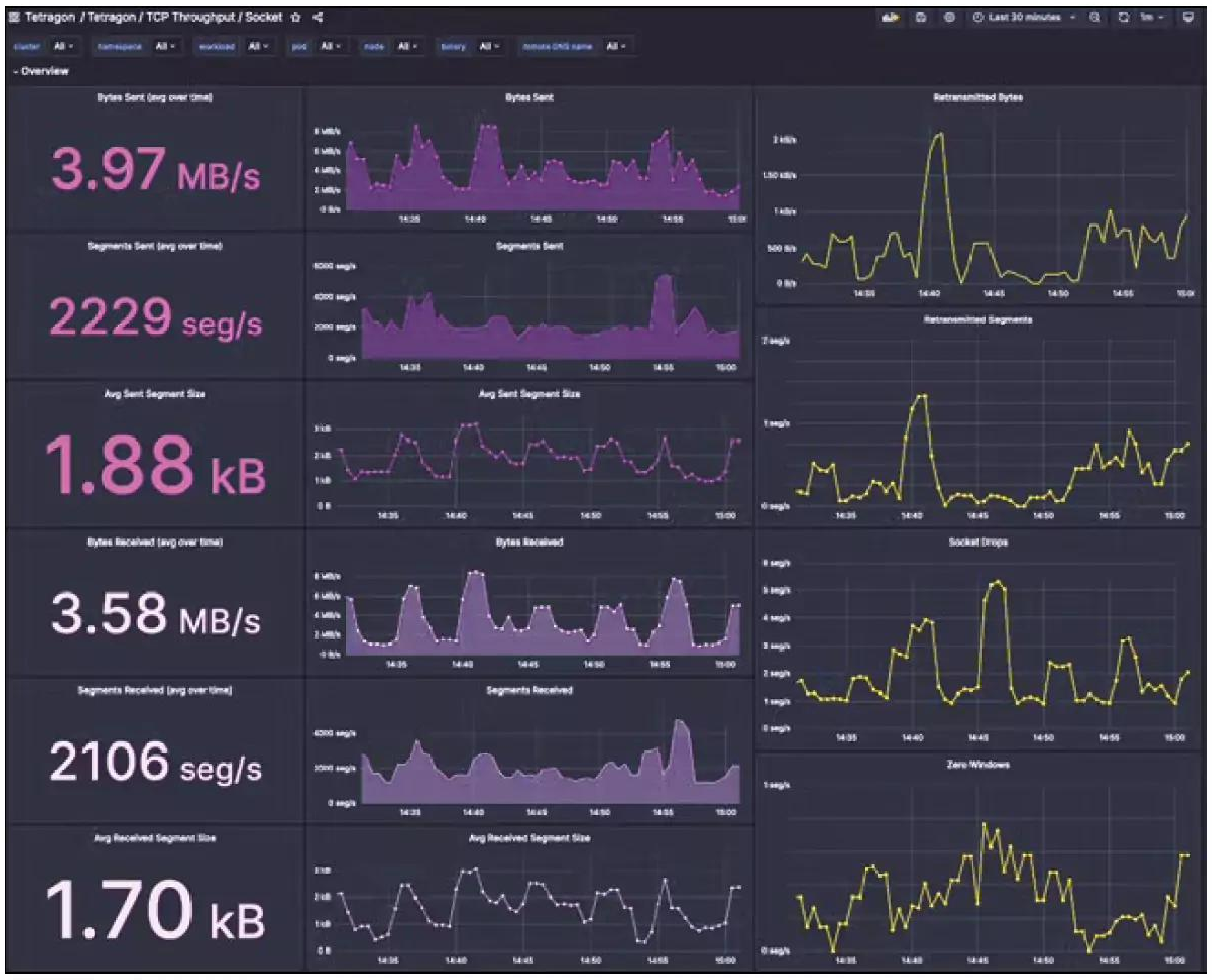

Tetragon / Tetragon / TCP Throughput / Socket
Last 30 minutes
Overview
Bytes Sent (avg over time)
3.97 MB/s
Bytes Sent
Retransmitted Bytes
Segments Sent (avg over time)
2229 seg/s
Segments Sent
Retransmitted Segments
Avg Sent Segment Size
1.88 kB
Avg Sent Segment Size
Bytes Received (avg over time)
3.58 MB/s
Bytes Received
Socket Drops
Segments Received (avg over time)
2106 seg/s
Segments Received
Zero Windows
Avg Received Segment Size
1.70 kB
Avg Received Segment Size

Figure 4: Tetragon can be connected automatically to Prometheus from inside Kubernetes. Dashboards by Cilium are available on the Grafana Marketplace. © Isovalent

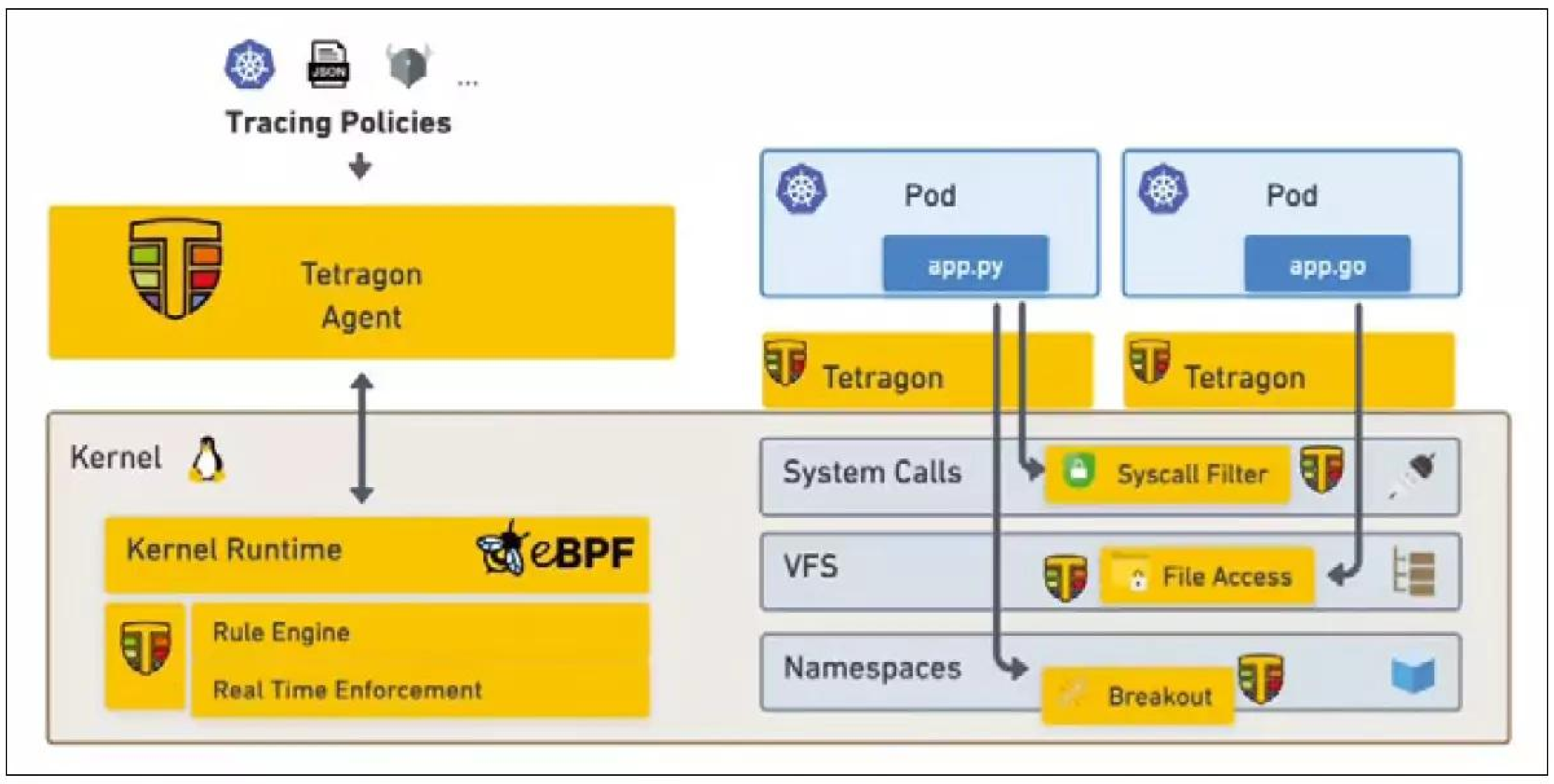

JSON
...
Tracing Policies
Tetragon Agent
Pod
app.py
Pod
app.go
Tetragon
Tetragon
Kernel
Kernel Runtime
eBPF
Rule Engine
Real Time Enforcement
System Calls
Syscall Filter
VFS
File Access
Namespaces
Breakout

Figure 5: Tetragon's tracing capabilities enable information tracking through a thicket of communication endpoints, as is often the case with microarchitectures. To do this, the tool taps into application communication deep within compute nodes. © Isovalent

Exporting Events

Far removed from classic MAT, Tetragon also offers the option of exporting events of any kind to the outside world. The tool has several interfaces for this purpose. On the one hand, Tetragon offers an interface for exporting raw event data in YAML format. However, this solution requires manual work because no ready-made parser function exists for this format, although output filters can at least be defined at the Tetragon level. This solution boosts performance and avoids the need to search for the right event notifications from among many.

If you don't want to evaluate the JSON interface yourself, why not let a suitable service (e.g., Fluentd and its JSON parser) do it for you? This arrangement also takes care of connections to SIEM systems. Tetragon offers an interface for gRPC, as well, and if you prefer to process the events at the command line, use the `tetra` CLI instead.

Conclusions

Tetragon is a comprehensive and complete tool designed specifically for workloads in which Kubernetes plays an important role. In contrast to Falco, it excels with its deep integration and abundance of well-designed features, although I was unable to cover all of them in detail in this article.

The comprehensive SDKs (**Figure 5**) are also worthy of mention; they offer direct integration with programs and empower Tetragon to implement comprehensive tracing on the basis of eBPF [7]. Considering the many examples and patterns for integrating Tetragon into running Kubernetes setups on the web, the tool is likely to be one of your first stops if you are looking for proactive security features for Kubernetes and need quickly actionable results. ■

Info

[1] "Proactive Threat Detection" by Martin Gerhard Loschwitz, *ADMIN*, issue 89, 2025, p. 44: [https://www.admin-magazine.com/Archive/2025/89/Proactive-threat-detection]

[2] Falco: [https://falco.org]

[3] Tetragon: [https://tetragon.io]

[4] Tetragon GitHub: [https://github.com/cilium/tetragon]

[5] Prometheus metrics: [https://tetragon.io/docs/installation/metrics/]

[6] Grafana dashboards for Tetragon: [https://grafana.com/orgs/isovalent/]

[7] Tracing in Tetragon: [https://tetragon.io/docs/concepts/tracing-policy/]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

What?!
I can get my issues SOONER?
Available anywhere, anytime!
ADMIN Network & Security
Automation
AD RMS
Cookie Dangers

Sign up for a digital subscription to improve your admin skills wth practical articles on network security, cloud computing, DevOps, HPC, storage and more!

Subscribe to the PDF edition:
shop.linuxnewmedia.com/shop

Secure AIX with Google Authenticator

Ready, Set, Auth

Modernizing AIX security with multifactor authentication has never been easier than with Google Authenticator, which brings two-factor authentication to your AIX system. By Chris Gibson

IBM AIX is a robust enterprise-grade operating system used in mission-critical environments where security is paramount, like banking, healthcare, and government. Security breaches on AIX systems are relatively rare compared with other operating systems such as Windows or Linux, primarily because of AIX's smaller attack surface, niche use in enterprise environments, and robust security architecture. However, breaches do occur, and their rarity doesn't mean AIX is immune.

Implementing two-factor authentication (2FA) or multifactor authentication (MFA) is a critical step in safeguarding AIX systems by helping to ensure that they remain secure against evolving threats and maintain compliance with regulatory demands.

AIX administrators have a few options to consider when it comes to MFA. One option is the IBM PowerSC product [1], which is a security tool that adds, among other things, MFA to IBM Power servers, requiring users to provide multiple forms of verification (like a password and a token) to access systems securely. It helps protect against unauthorized access and ensures compliance with security standards. Because it's included as part of the AIX Enterprise Edition bundle, if you have an active AIX Enterprise Edition license with software maintenance (SWMA), you might already be entitled to use PowerSC without additional licensing.

Another option is MFA with RSA SecurID [2]. This third-party solution typically comes with a price tag, requiring licenses for the RSA authentication manager software, user tokens, and potentially additional hardware or cloud services, depending on the deployment. Other options are available from other security software vendors that could be considered as well, such as Gemalto SafeNet Authentication Service. However, other solutions come with less cost and complexity that might be suitable for many environments. One such solution is Google Authenticator.

Google Authenticator for AIX

In October 2024, IBM announced that the Google Authenticator pluggable authentication module (PAM) for AIX was available [3]. This module enables 2FA for logging in to servers over SSH, OpenVPN, and the like, with both a time-based one-time password (TOTP) and hash-based message authentication code (HMAC) one-time password (HOTP) used as a second factor. The module is available for AIX at no charge from the IBM AIX Toolbox for Open Source Software public repository. The module is supported on AIX version 7.1 and higher.

TOTP is a security method that generates a unique, temporary code typically valid for 30 seconds and is used alongside a password for 2FA. It's like a constantly changing PIN that you get from an app like Google Authenticator to prove it's really you logging in.

The HOTP security method generates a unique code for 2FA with a counter; that code is valid until the counter increments. HMAC is a unique, temporary code generated by a secret key and a counter, valid for a single use to enhance security in authentication systems. It's like a one-time PIN from an app that changes each time you use it, ensuring secure logins.

Although traditional password-based authentication remains a common practice for securing systems like AIX, it is increasingly vulnerable to modern threats. Here, I address that challenge by detailing the process of installing

Lead Image © alphaspirit, 123RF.com

and configuring the Google Authenticator PAM module on AIX, providing a robust second layer of security for your OpenSSH environment with the use of both TOTP and HOTP.

2FA on AIX with the Google PAM Module

To begin, use DNF to install the *google-authenticator* package on the AIX host. In the examples that follow, I configure the module for AIX 7.3. To do this you'll need to have the AIX Toolbox already configured on your AIX system. If you're new to the toolbox and need help installing and configuring DNF on your AIX hosts, I recommend you review a previous article on managing FOSS applications on AIX [4]. If you'd also like some hands-on experience with DNF on AIX, be sure to check out the available training on this topic [5].
To install the *google-authenticator* package and check that the package is installed, run the commands

```
dnf -y install google-authenticator
rpm -q google-authenticator
```

You should see a package named *google-authenticator-1.10-1.ppc* installed on your system, which provides the required `pam_google_authenticator.so` PAM. If `dnf` is not an option in your environment, you can try working around it with the commands

```
wget
--content-disposition ➘
       https://ibm.biz/aixmfagoogle
rpm -Uvh google-authenticator-➘
         1.10-1.aix7.1.ppc.rpm
```

The first command downloads the Google Authenticator RPM package directly from the AIX Toolbox site and the second installs it.

Configure pam.conf

The next step is to configure the `/etc/pam.conf` file to enable PAM as an SSH authentication method and use the Google Authenticator module by adding several new entries to this file [6]. Always back up a configuration file first before making changes to it.
Add the lines shown in Listing 1 to the bottom of the `/etc/pam.conf` file, save it, and exit your editor.

Listing 1: `/etc/pam.conf`

```
sshd auth required  pam_aix
sshd auth required /usr/lib/security/pam_google_authenticator.so nullok debug
  no_increment_hotp noskewadj
sshd account required pam_aix
sshd password required pam_aix
sshd session required pam_aix
```

If the long `sshd` command with the `nullok` entry is not entered all on one line (e.g., if it is split over two lines) – particularly if the `nullok` option is either missing, on a different line, or entered incorrectly – users (even the root user) will likely be prevented from logging in to the server if they have not been configured for the *google-authenticator* package.
In this setup, `pam_aix` refers to the PAM for AIX, which integrates AIX's native authentication, account management, password handling, and session management into the PAM framework for SSH. It uses the AIX system mechanisms (like `/etc/passwd` or `/etc/security`) to verify user credentials, account status, passwords, and session setup. Table 1 provides a brief description of each Google Authenticator entry and its purpose.
The `pam_google_authenticator.so` module provides many options for different types of login configurations. Table 2 lists the options chosen to use with the module and describes their functions. For a complete list and description of the available options, please refer to the official GitHub repo page [7].

Table 1: Entries in `/etc/pam.conf`

Entry	Description
`sshd auth required pam_aix`	Handles user authentication for SSH with AIX's built-in authentication mechanisms, like checking passwords against the system database
`sshd auth required /usr/lib/security/pam_google_authenticator.so nullok debug no_increment_hotp noskewadj`	Adds two-factor authentication for SSH with Google Authenticator, allowing optional one-time codes, debug logging, and specific time-based OTP settings
`sshd account required pam_aix`	Verifies user account status for SSH, ensuring the account is valid and not locked or expired
`sshd password required pam_aix`	Manages password validation and updates for SSH logins with AIX password policies
`sshd session required pam_aix`	Sets up the user session for SSH, handling tasks like setting environment variables and logging session activity

Table 2: Google Authenticator Module Options

Option	Description
`debug`	Enables verbose logging to `syslog`, providing detailed information about the authentication process for troubleshooting
`no_increment_hotp`	Doesn't increment the counter for failed HOTP attempts
`nullok`	Allows users to log in without a one-time password (OTP), if they haven't set up OTP yet
`noskewadj`	Disables automatic compensation for time differences between the server and the device generating one-time codes, requiring precise time synchronization for TOTP authentication to work

Enable PAM Authentication

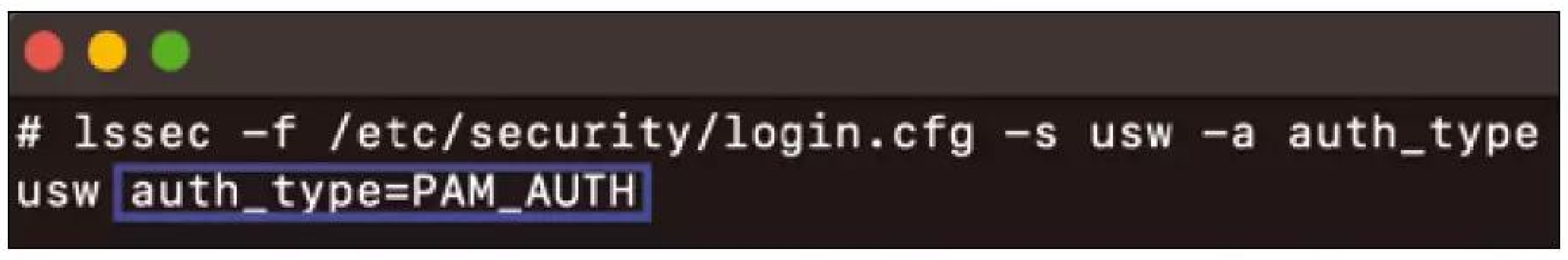
To enable PAM authentication, the `/etc/security/login.cfg` file needs to be modified. Change `auth_type` from `STD_AUTH` to `PAM_AUTH` with

```
chsec -f /etc/security/login.cfg ➥
      -s usw -a auth_type=PAM_AUTH
```

Confirm that `auth_type` has been set correctly by running the `lssec` command shown in Figure 1.

Configure OpenSSH to Use PAM

Edit the `/etc/ssh/sshd_config` file and change `UsePAM` from `no` to `yes` (`UsePAM=yes`) to enable PAM authentication for SSH. If the entry is missing from your `sshd_config` file, add it now and set it to `yes`.



```
# lssec -f /etc/security/login.cfg -s usw -a auth_type
usw auth_type=PAM_AUTH
```

Figure 1: Confirming that `auth_type` is set to `PAM_AUTH` in `login.cfg`.

Confirm that `UsePAM` is set to `yes` in `sshd_config`:

```
grep UsePAM /etc/ssh/sshd_config
UsePAM yes
```

After making the change to the `sshd_config` file, restart the `sshd` subsystem and confirm that `sshd` has restarted successfully:

```
stopsrc -s sshd ; sleep 5 ; startsrc -s sshd
lssrc -s sshd
```

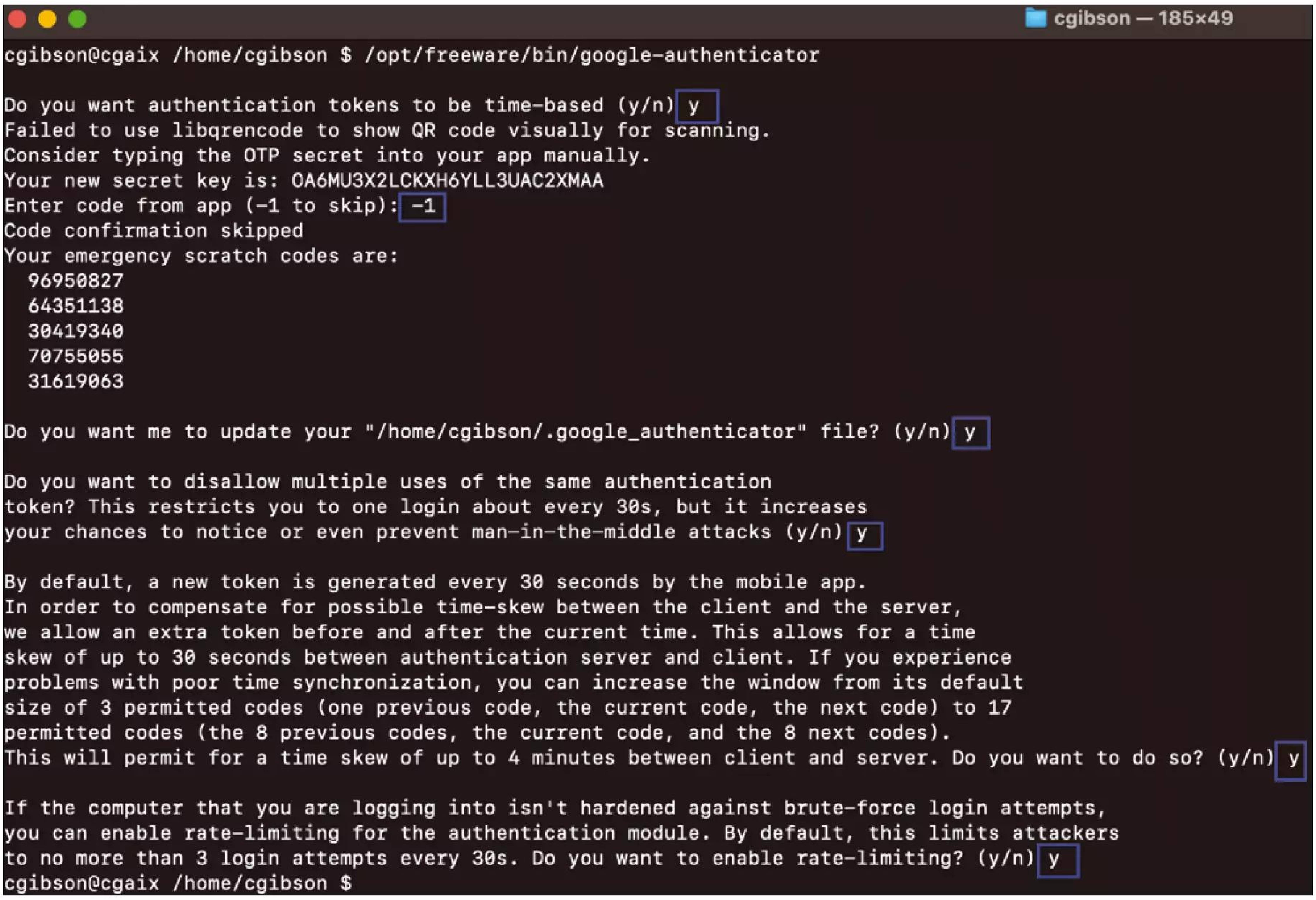
Its status should report as *active*.

AIX User Setup for TOTP

The Google Authenticator setup, for both TOTP and HOTP configurations, can be performed on a per-user basis. You don't need to configure TOTP/HOTP login verification for all users. Instead, you can configure it for only specific users.

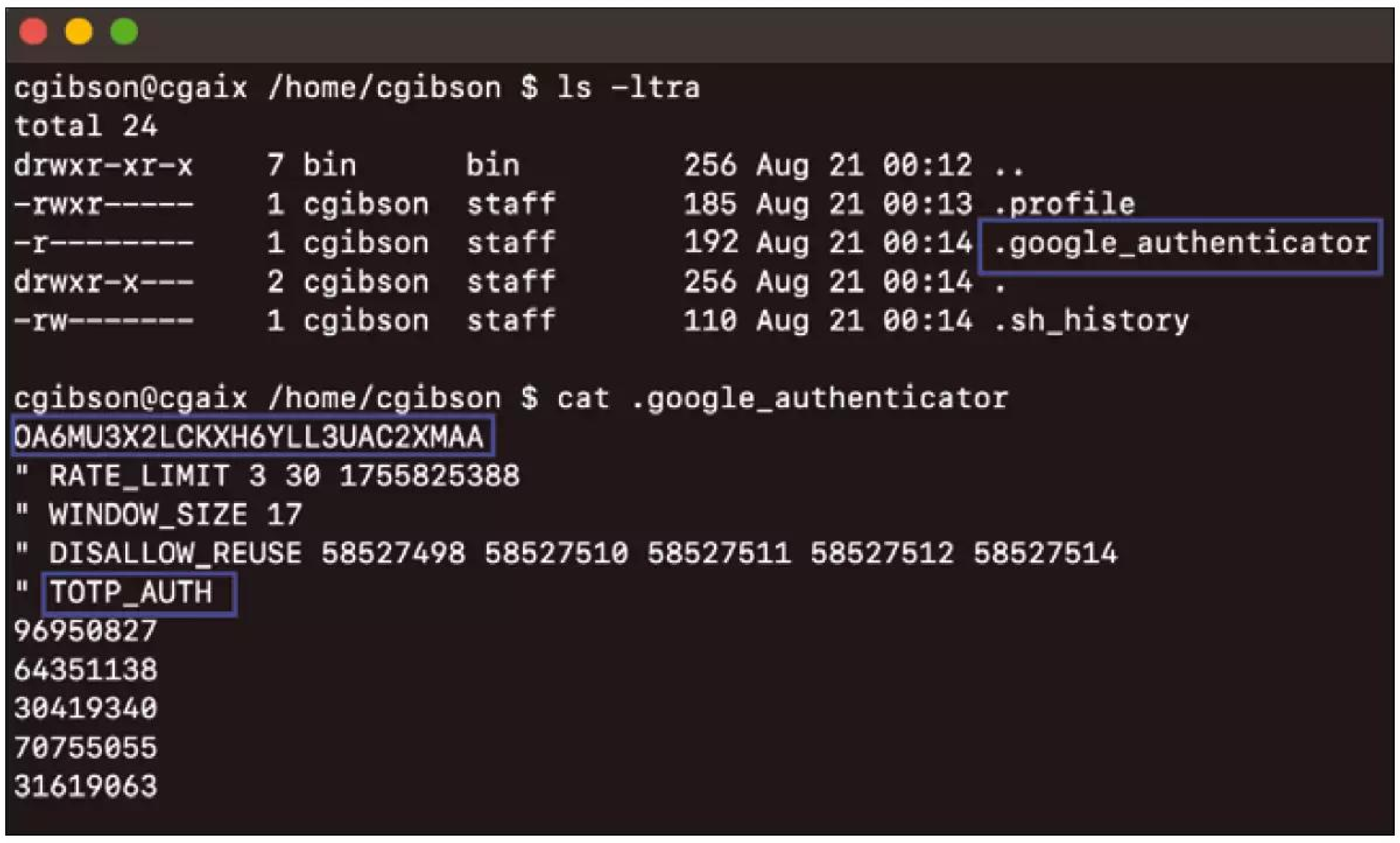
In this example. I'll show you how to set up Google Authenticator, assuming a username *cgibson*, on the AIX server. As the user, you run the `/opt/freeware/bin/google-authenticator` tool to start the TOTP setup process. Several questions are asked in the setup process (Figure 2):

- *Do you want authentication tokens to be time-based (y/n)*: Answer *y* to set up TOTP specifically.
- *Enter code from app*: Enter *-1* to skip this step.
- *Do you want me to update your "/home/cgibson/.google_authenticator" file?*: Enter *y* to create a new file.
- *Do you want to disallow multiple uses of the same authentication token?*: Respond *y* to prevent the same authentication token from being reused.
- *By default, a new token is generated every 30 seconds by the mobile app …. Do you want to do so?*: Enter *y* to account for potential time synchronization issues between the mobile device and the server.



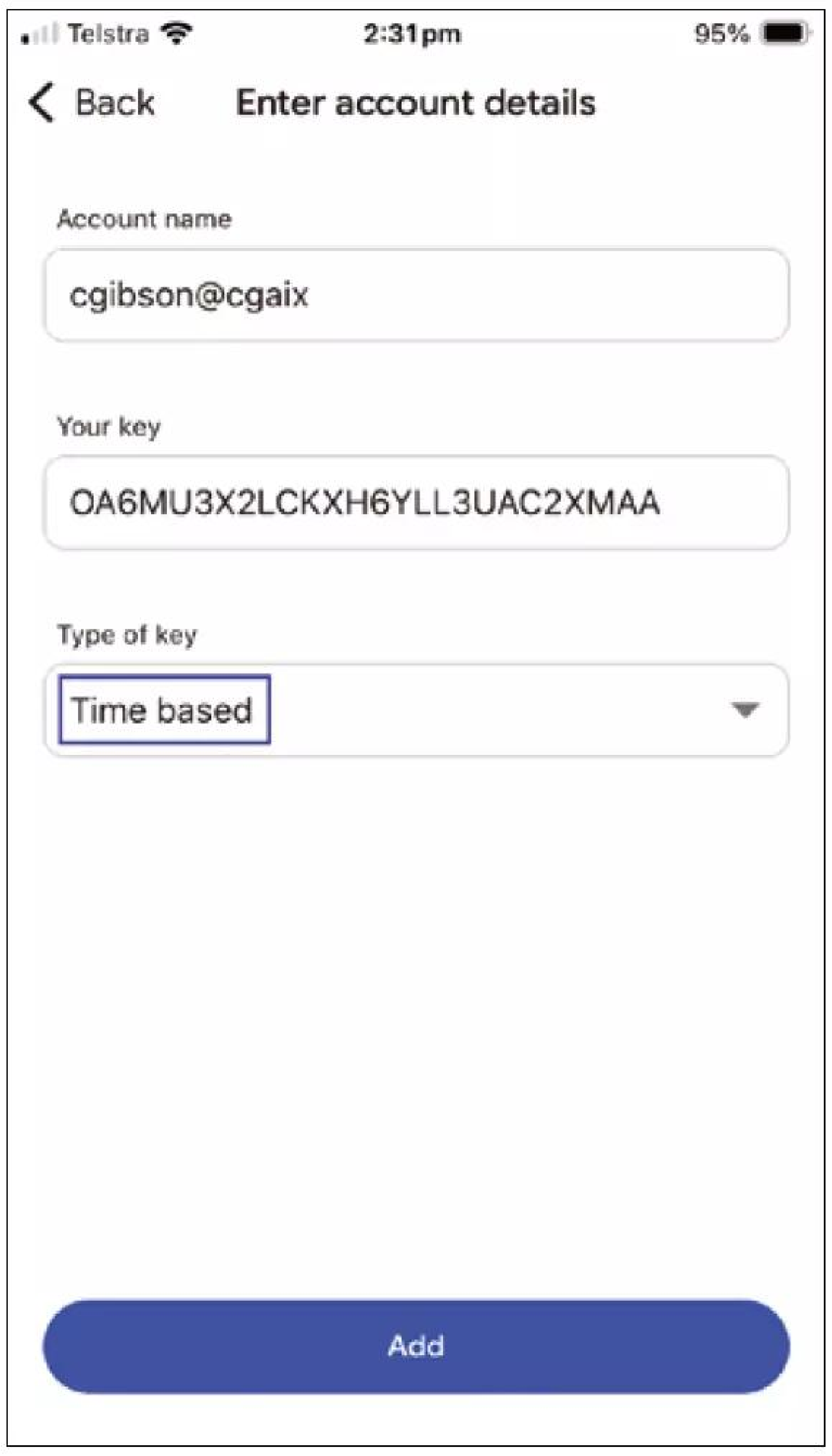
```
cgibson@cgaix /home/cgibson $ /opt/freeware/bin/google-authenticator

Do you want authentication tokens to be time-based (y/n) y
Failed to use libqrencode to show QR code visually for scanning.
Consider typing the OTP secret into your app manually.
Your new secret key is: OA6MU3X2LCKXH6YLL3UAC2XMAA
Enter code from app (-1 to skip): -1
Code confirmation skipped
Your emergency scratch codes are:
  96950827
  64351138
  30419340
  70755055
  31619063

Do you want me to update your "/home/cgibson/.google_authenticator" file? (y/n) y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a time
skew of up to 30 seconds between authentication server and client. If you experience
problems with poor time synchronization, you can increase the window from its default
size of 3 permitted codes (one previous code, the current code, the next code) to 17
permitted codes (the 8 previous codes, the current code, and the 8 next codes).
This will permit for a time skew of up to 4 minutes between client and server. Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-force login attempts,
you can enable rate-limiting for the authentication module. By default, this limits attackers
to no more than 3 login attempts every 30s. Do you want to enable rate-limiting? (y/n) y
cgibson@cgaix /home/cgibson $
```

Figure 2: Configuring TOTP for AIX user *cgibson*.



```
cgibson@cgaix /home/cgibson $ ls -ltra
total 24
drwxr-xr-x    7 bin      bin          256 Aug 21 00:12 ..
-rwxr-----    1 cgibson  staff        185 Aug 21 00:13 .profile
-r--------    1 cgibson  staff        192 Aug 21 00:14 .google_authenticator
drwxr-x---    2 cgibson  staff        256 Aug 21 00:14 .
-rw-------    1 cgibson  staff        110 Aug 21 00:14 .sh_history

cgibson@cgaix /home/cgibson $ cat .google_authenticator
OA6MU3X2LCKXH6YLL3UAC2XMAA
" RATE_LIMIT 3 30 1755825388
" WINDOW_SIZE 17
" DISALLOW_REUSE 58527498 58527510 58527511 58527512 58527514
" TOTP_AUTH
96950827
64351138
30419340
70755055
31619063
```

Figure 3: The `.google_authenticator` file with TOTP setup.



Account name
cgibson@cgaix

Your key
OA6MU3X2LCKXH6YLL3UAC2XMAA

Type of key
Time based

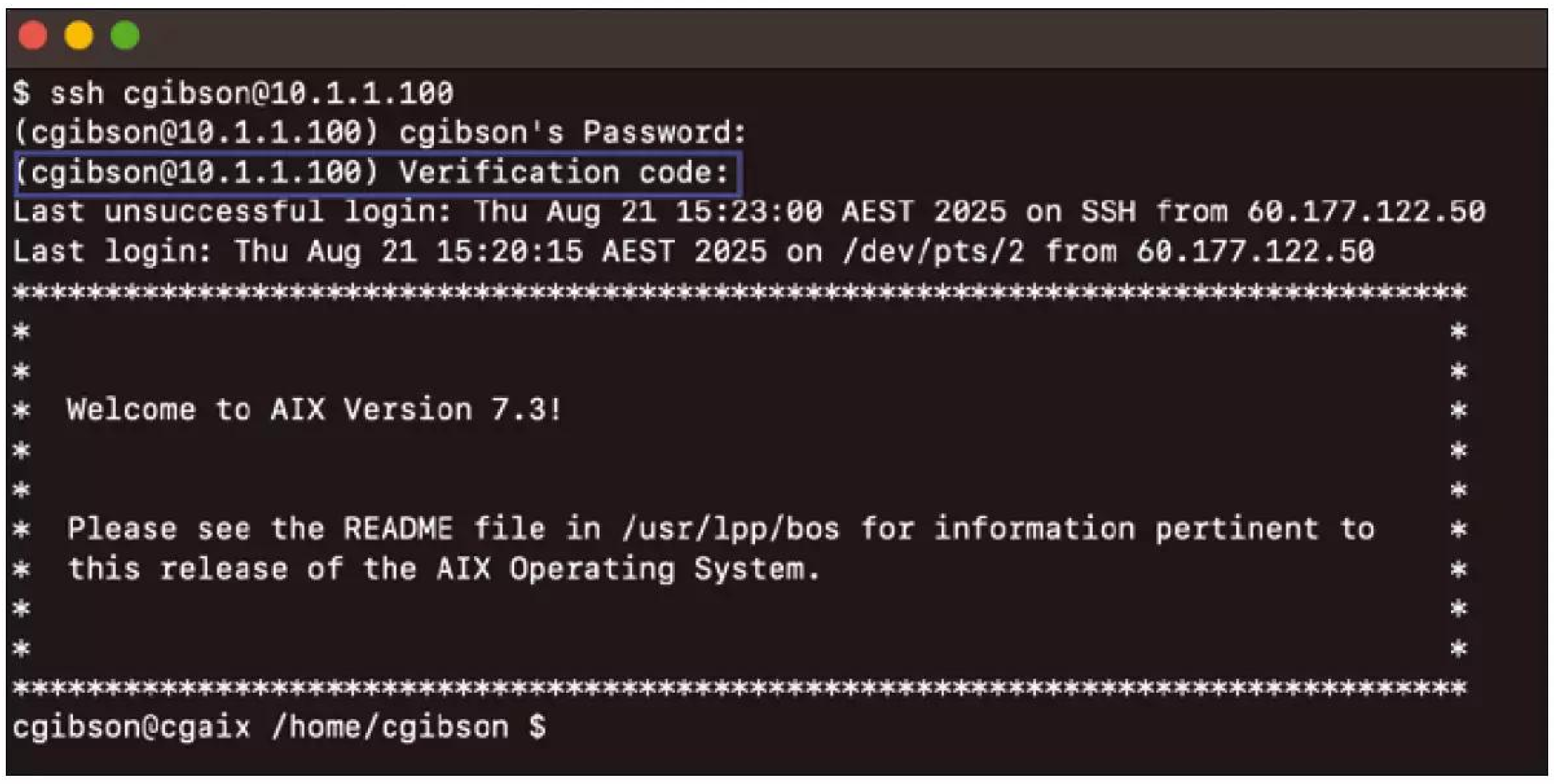
Figure 4: Adding the time-based (TOTP) key in the Google Authenticator app.

- *Do you want to enable rate-limiting?*: Respond *y* to limit login attempts to three every 30 seconds to protect against brute-force attacks.

In your home directory, a new file named `.google_authenticator` is created that contains the configuration details for the TOTP setup. The most important piece of information in this file is the secret key (the alphanumeric sequence outlined in blue in Figure 3). When configuring TOTP, you need to enter this key into the Google Authenticator app on your personal mobile device (typically a smartphone).

Setup for TOTP

Now, start the Google Authenticator app on the smartphone. In the bottom-right corner of the app, tap the multicolored plus symbol and select *Enter a setup key*. Enter an account name (e.g., *cgibson@cgaix*; Figure 4) for this new TOTP setup and add the secret key from your `.google_authenticator` file. Under *Type of key*, ensure that *Time based* is selected. Tap *Add* when you're done.



```
$ ssh cgibson@10.1.1.100
(cgibson@10.1.1.100) cgibson's Password:
(cgibson@10.1.1.100) Verification code:
Last unsuccessful login: Thu Aug 21 15:23:00 AEST 2025 on SSH from 60.177.122.50
Last login: Thu Aug 21 15:20:15 AEST 2025 on /dev/pts/2 from 60.177.122.50
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 7.3!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
cgibson@cgaix /home/cgibson $
```

Figure 5: User SSH session entering TOTP verification code for `google-authenticator`.

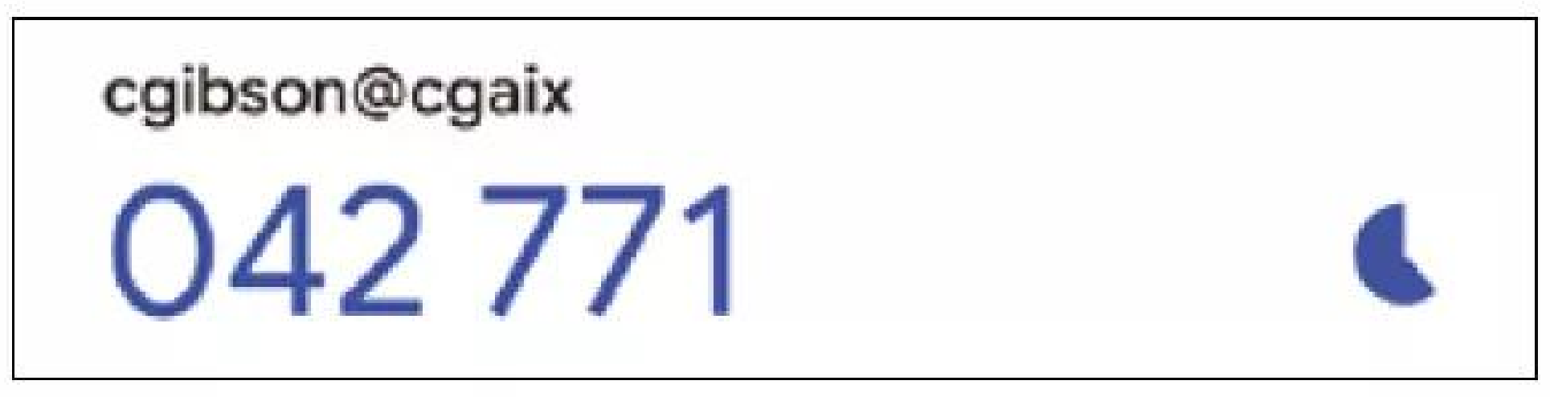
SSH Login with TOTP

With the Google Authenticator setup complete, you can now test it by logging in with your password and TOTP code (from the Google Authenticator app). To open a new SSH session with the AIX server, run the `ssh` command from the local workstation. The first prompt asks for the local user password (this is your password set on the AIX host). After successfully entering the password, you are then prompted to enter your *Verification code* (Figure 5, blue box). Now open the Google Authenticator app on your personal device and locate the code displayed under the *cgibson@cgaix* (in this example) account in the app (Figure 6); enter this code into the login session. With the correct code entered, you are then logged in to the AIX server and presented with a user shell environment.

AIX User Setup for HOTP

The next example sets up the same user, *cgibson*, except this time you'll configure HOTP. The `google-authenticator` utility is run once more (Figure 7), with several questions to be answered once again:

- *Do you want authentication tokens to be time-based (y/n)*: To specifically configure HOTP, enter *n*.
- *Do you want me to update your "/home/cgibson/.google_authenticator" file?*: Enter *y*.
- *By default, three tokens are valid at any one time … this window can be increased from its default size of 3 to 17. Do you want to do so?*: Respond *y*.



cgibson@cgaix
042 771

Figure 6: Displaying the TOTP code for the user in the Google Authenticator app.

- *Do you want to enable rate-limiting?*: Enter *y*.

In your home directory a file named `.google_authenticator` is created anew; however, this time it contains the configuration details for an HOTP setup (Figure 8). Just like the TOTP setup, the HOTP secret key is saved to this file and must be entered into the Google Authenticator app.

Setup for HOTP

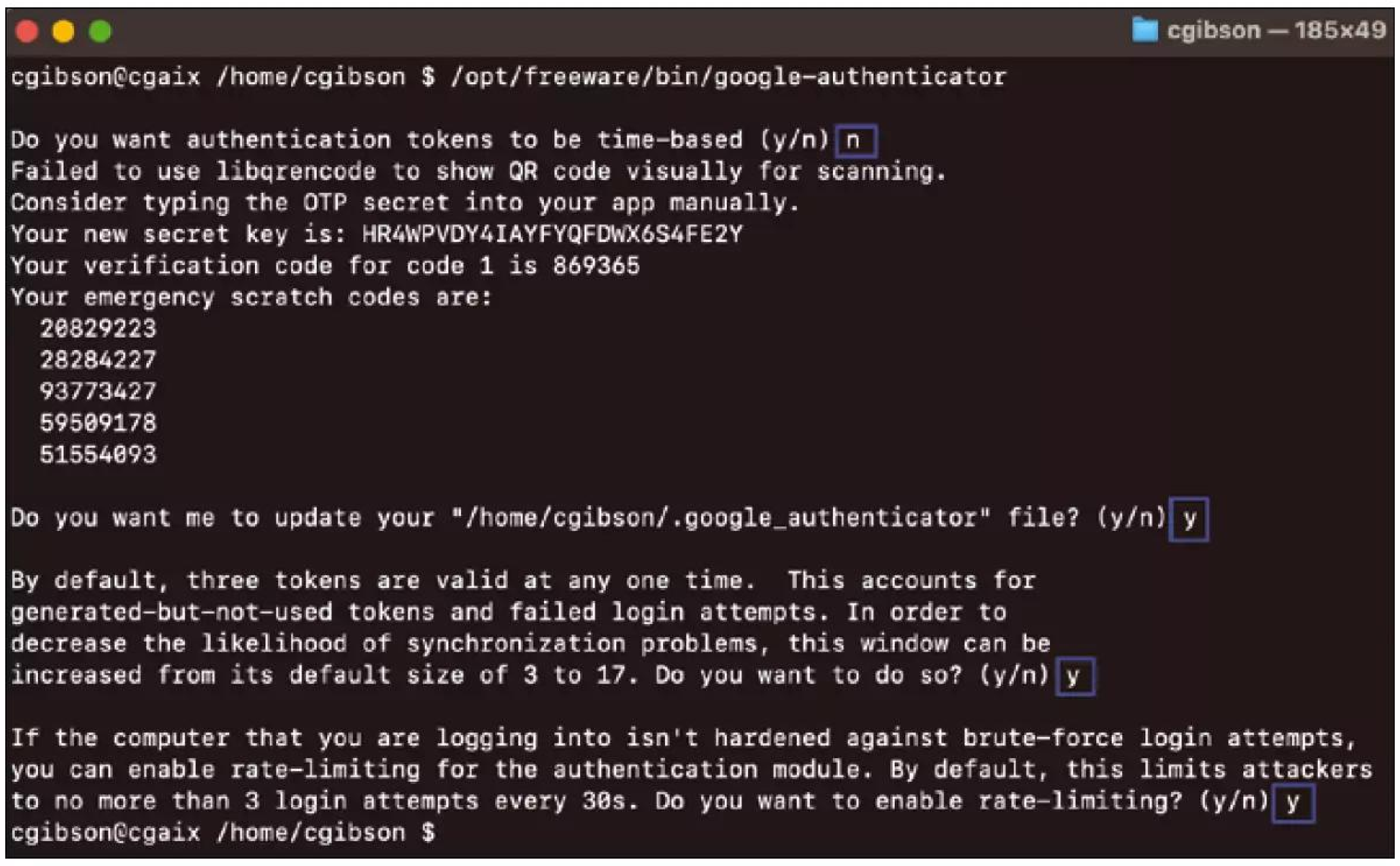
Once again the Google Authenticator app needs to be started on the personal mobile device. Tap the multicoloured plus symbol and select *Enter a setup key*. The account name for the new HOTP configuration (e.g., *cgibson@cgaix HOTP*) is entered and then the secret key is input in the appropriate field. For HOTP you must ensure that *Counter based* is selected under *Type of key* instead of *Time based*, as before for TOTP. Tap *Add* when done.

SSH Login with HOTP

With the HOTP user setup complete, you can now test it by logging in with your password and the HOTP code from the app. To begin, start an SSH session with the AIX host; after entering your password, you're prompted for your *Verification code*. Once again, you open the Google Authenticator app on your personal device and locate the HOTP code displayed under the *cgibson@cgaix HOTP* account (Figure 9). You might need to refresh the code by tapping on the reload icon. Now enter the HOTP code and you're immediately logged in to the AIX server. Note that the session details look exactly like the TOTP login session in Figure 5; there's nothing to indicate which type of OTP is required or was used.

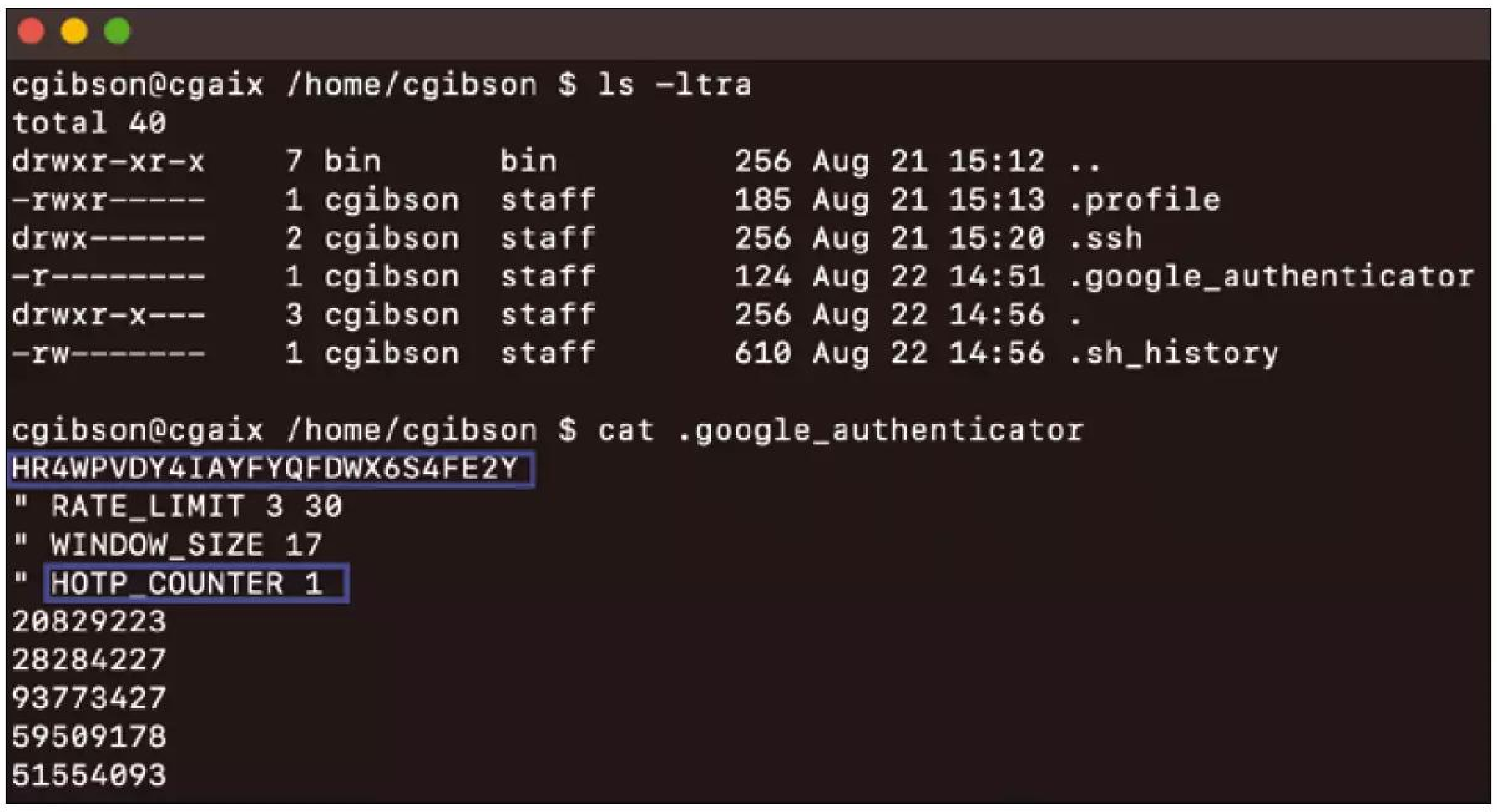
In the unlikely event that you need to use (switch between) a combination of TOTP and HOTP code usage, you can run the `google-authenticator` tool twice: once for TOTP and then again for HOTP. After each run, you can copy the resulting `.google_authenticator` file to `.google_authenticator.totp` for TOTP and to `.google_authenticator.hotp` for HOTP:

```
cp -p .google_authenticator ➘
     .google_authenticator.totp
```



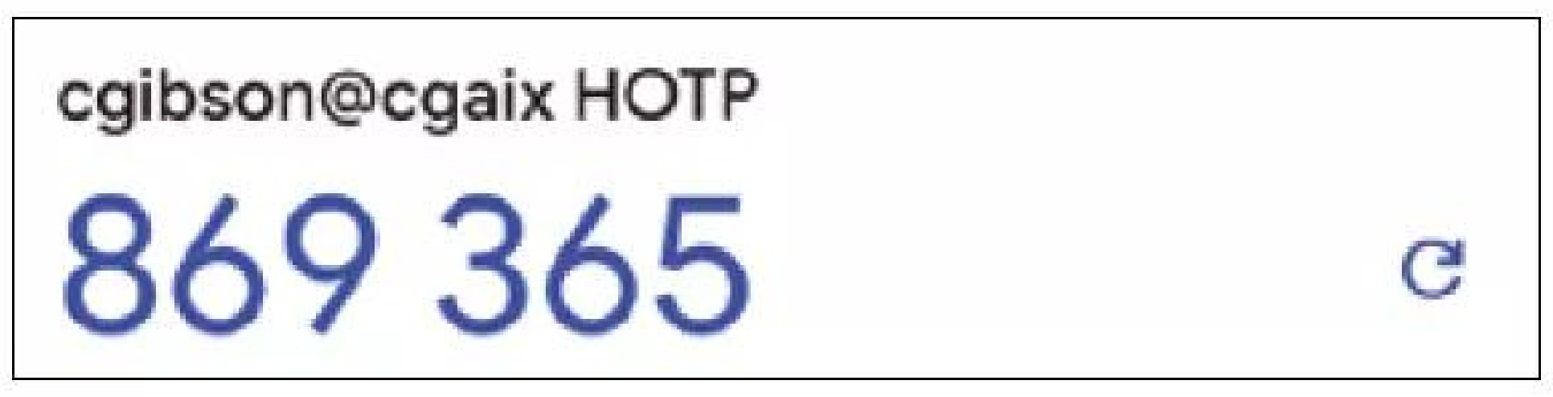
```
cgibson — 185x49
cgibson@cgaix /home/cgibson $ /opt/freeware/bin/google-authenticator

Do you want authentication tokens to be time-based (y/n) n
Failed to use libqrencode to show QR code visually for scanning.
Consider typing the OTP secret into your app manually.
Your new secret key is: HR4WPVDY4IAYFYQFDWX6S4FE2Y
Your verification code for code 1 is 869365
Your emergency scratch codes are:
  20829223
  28284227
  93773427
  59509178
  51554093

Do you want me to update your "/home/cgibson/.google_authenticator" file? (y/n) y

By default, three tokens are valid at any one time.  This accounts for
generated-but-not-used tokens and failed login attempts. In order to
decrease the likelihood of synchronization problems, this window can be
increased from its default size of 3 to 17. Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-force login attempts,
you can enable rate-limiting for the authentication module. By default, this limits attackers
to no more than 3 login attempts every 30s. Do you want to enable rate-limiting? (y/n) y
cgibson@cgaix /home/cgibson $
```

Figure 7: Configuring HOTP for AIX user *cgibson*.



```
cgibson@cgaix /home/cgibson $ ls -ltra
total 40
drwxr-xr-x    7 bin      bin          256 Aug 21 15:12 ..
-rwxr-----    1 cgibson  staff        185 Aug 21 15:13 .profile
drwx------    2 cgibson  staff        256 Aug 21 15:20 .ssh
-r--------    1 cgibson  staff        124 Aug 22 14:51 .google_authenticator
drwxr-x---    3 cgibson  staff        256 Aug 22 14:56 .
-rw-------    1 cgibson  staff        610 Aug 22 14:56 .sh_history

cgibson@cgaix /home/cgibson $ cat .google_authenticator
HR4WPVDY4IAYFYQFDWX6S4FE2Y
" RATE_LIMIT 3 30
" WINDOW_SIZE 17
" HOTP_COUNTER 1
20829223
28284227
93773427
59509178
51554093
```

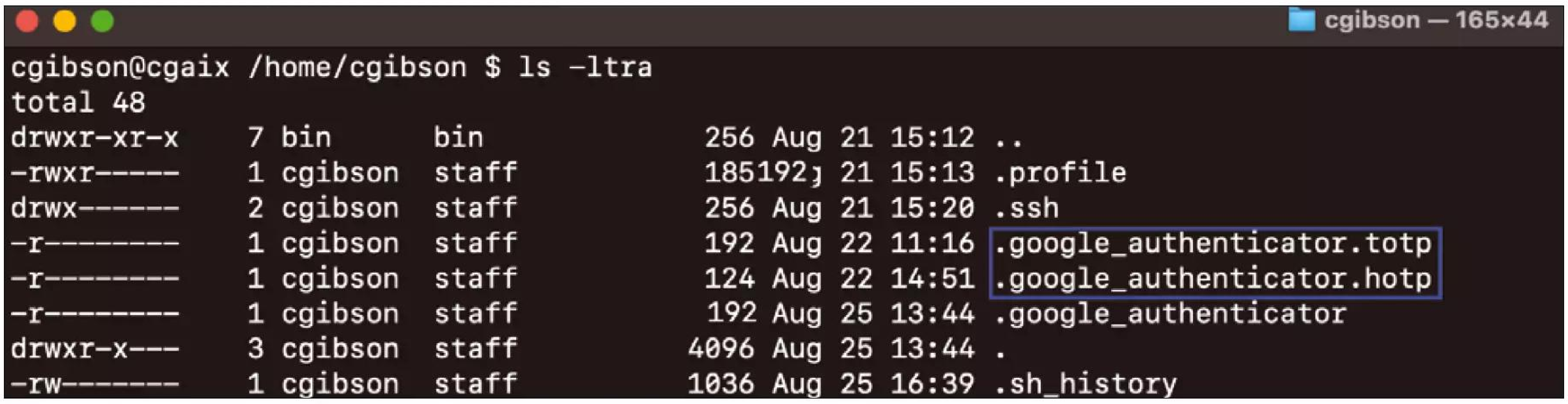
Figure 8: The `.google_authenticator` file with HOTP setup.



cgibson@cgaix HOTP

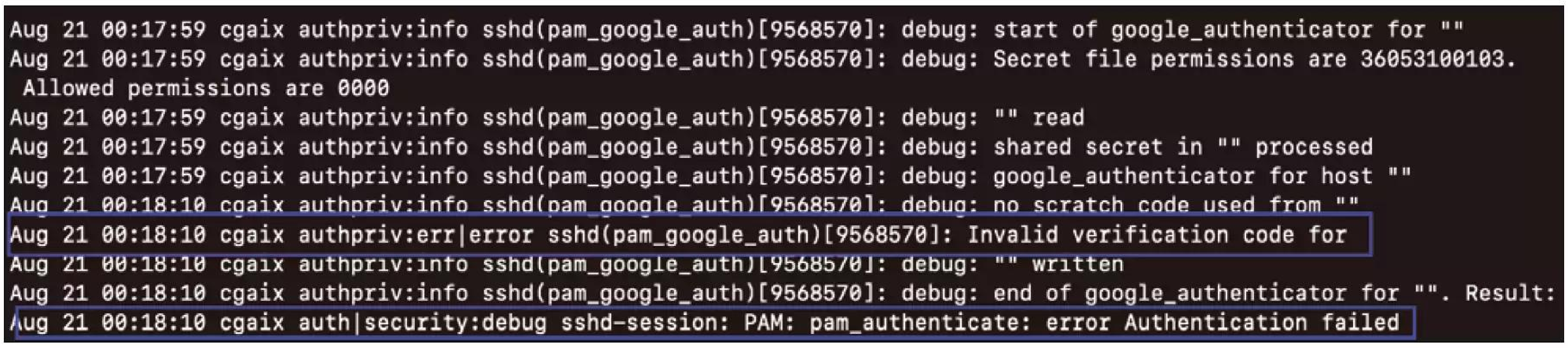
869 365

Figure 9: The user HOTP code displayed in the Google Authenticator app.



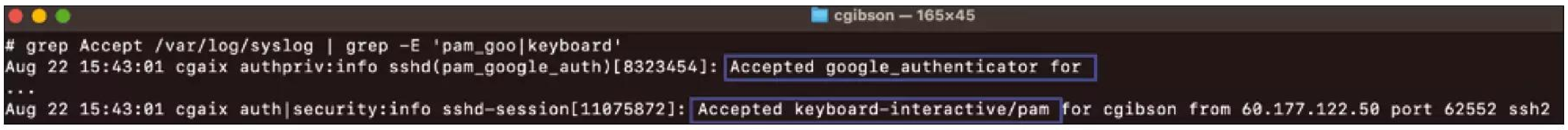
```
cgibson — 165x44
cgibson@cgaix /home/cgibson $ ls -ltra
total 48
drwxr-xr-x    7 bin      bin            256 Aug 21 15:12 ..
-rwxr-----    1 cgibson  staff          185192j 21 15:13 .profile
drwx------    2 cgibson  staff          256 Aug 21 15:20 .ssh
-r--------    1 cgibson  staff          192 Aug 22 11:16 .google_authenticator.totp
-r--------    1 cgibson  staff          124 Aug 22 14:51 .google_authenticator.hotp
-r--------    1 cgibson  staff          192 Aug 25 13:44 .google_authenticator
drwxr-x---    3 cgibson  staff         4096 Aug 25 13:44 .
-rw-------    1 cgibson  staff         1036 Aug 25 16:39 .sh_history
```

Figure 10: Both TOTP and HOTP `.google_authenticator` files are available.



```
Aug 21 00:17:59 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: start of google_authenticator for ""
Aug 21 00:17:59 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: Secret file permissions are 36053100103.
 Allowed permissions are 0000
Aug 21 00:17:59 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: "" read
Aug 21 00:17:59 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: shared secret in "" processed
Aug 21 00:17:59 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: google_authenticator for host ""
Aug 21 00:18:10 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: no scratch code used from ""
Aug 21 00:18:10 cgaix authpriv:err|error sshd(pam_google_auth)[9568570]: Invalid verification code for
Aug 21 00:18:10 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: "" written
Aug 21 00:18:10 cgaix authpriv:info sshd(pam_google_auth)[9568570]: debug: end of google_authenticator for "". Result:
Aug 21 00:18:10 cgaix auth|security:debug sshd-session: PAM: pam_authenticate: error Authentication failed
```

Figure 11: Syslog output for invalid Google Authenticator code.



```
# grep Accept /var/log/syslog | grep -E 'pam_goo|keyboard'
Aug 22 15:43:01 cgaix authpriv:info sshd(pam_google_auth)[8323454]: Accepted google_authenticator for
...
Aug 22 15:43:01 cgaix auth|security:info sshd-session[11075872]: Accepted keyboard-interactive/pam for cgibson from 60.177.122.50 port 62552 ssh2
```

Figure 12: Syslog output for successful `pam_google_auth` module with SSH session.

```
cp -p .google_authenticator ↲
      .google_authenticator.hotp
```

Whenever you need to use either TOTP or HOTP you can simply copy the appropriate file over the top of the `.google_authenticator` file (Figure 10).

SSH and PAM Syslog Logging

To assist with troubleshooting OpenSSH login issues (with Google Authenticator PAM), you can configure additional logging on the AIX host [8]. To enable PAM debugging, create an empty file in `/etc`:

```
touch /etc/pam_debug
```

Enable `syslogd` to capture debug output by adding the entry

```
*.debug /var/log/syslog rotate size ↲
  50m files 10 compress
```

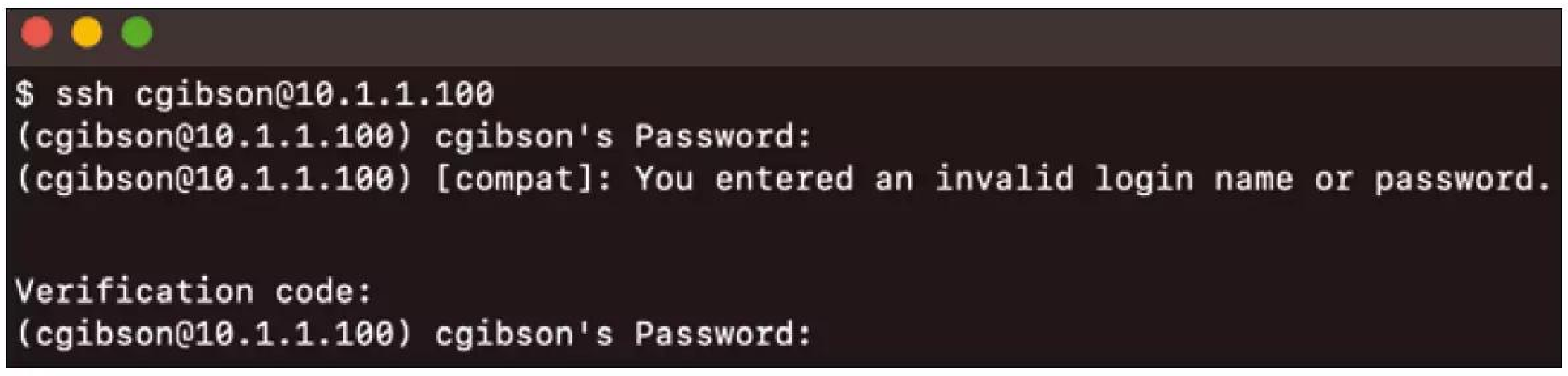
to the `/etc/syslog.conf` file. The AIX `syslog` entry directs all debug-level messages to the `/var/log/syslog` file, rotating it when it reaches 50MB, keeping up to 10 archived files, and compressing them to save space. Next edit the `/etc/ssh/sshd_config` file and enable logging of debug data. Under the `#Logging` section of the file, add `LogLevel DEBUG3` to the `sshd_config` file; then, restart the `syslogd` daemon and the `sshd` subsystem:

```
stopsrc -s syslogd ; ↲
  sleep 10; startsrc -s syslogd
stopsrc -s sshd; sleep 10 ; ↲
  startsrc -s sshd
```

With the additional logging in place you can use it to help diagnose issues with the Google Authenticator PAM configuration. For example, you can search the `/var/log/syslog` logfile for *pam_google_auth*. In Figure 11, you can see that the user was unable to log in with their TOTP code because the date and time were set incorrectly on the AIX host (explained further in the "TOTP Code Doesn't Work" section). Once this was corrected (i.e., the date and time matched the user's time zone), the user was then able to log in successfully (not that you can discern this from the available `syslog` data, but it can help in pinpointing when an issue is occurring). The `syslog` output in Figure 12 is an example of a successful login session over SSH with `pam_google_auth`. You can use `grep` to search for the string *Accept*. A correct and valid AIX user password must be entered before you can enter the required TOTP/HOTP verification code. If you enter your password incorrectly you'll see a message stating you've entered an invalid username or password. Even if you then enter a valid verification code from the app, you'll be prompted to re-enter your password and start over (Figure 13).

TOTP Code Doesn't Work

One issue that can often arise and catch you out, specifically with TOTP, is server clock issues. TOTP codes generated by Google Authenticator rely heavily on accurate time synchronization between the client device where the app is installed and the server where the TOTP is being validated. Issues with time (the clock) on an AIX server can lead to invalid TOTP codes and failed authentication. If the time (and date) on your AIX server and your Google Authenticator device are significantly different, it can prevent TOTP codes from allowing you to log in to your server over SSH. TOTP systems typically allow a small time discrepancy (e.g., ±30 seconds). However, larger differences (such as minutes or hours) will cause failures. If TOTP authentication fails and you cannot log in, you might need alternative access to the server (e.g., from a console, root access, an alternative user account, a backup SSH key, etc.) to resolve the issue.



```
$ ssh cgibson@10.1.1.100
(cgibson@10.1.1.100) cgibson's Password:
(cgibson@10.1.1.100) [compat]: You entered an invalid login name or password.

Verification code:
(cgibson@10.1.1.100) cgibson's Password:
```

Figure 13: User entered incorrect password and must start the Google Authenticator process again.

Ensure your AIX server uses the Network Time Protocol (NTP, `xntpd`) to synchronize with a reliable time source, and verify that your Google Authenticator device (your smartphone/device) has automatic time sync enabled.

For specific information on this problem and how to handle it, refer to the Google GitHub `google-authenticator-libpam` page **[9]**, which states:

> *If you discover that your TOTP code never works, this is most commonly the result of the clock on your server being different from the one on your Android device. The PAM module makes an attempt to compensate for time skew. You can teach it about the amount of skew that you are experiencing by trying to log it three times in a row. Make sure you always wait 30s (but not longer) so that you get three distinct TOTP codes. Some administrators prefer that time skew isn't adjusted automatically, as doing so results in a slightly less secure system configuration. If you want to disable it, you can do so on the module command line:* `auth required pam_google_authenticator.so noskewadj`

TOTP or HOTP?

TOTP is generally recommended over HOTP when using the Google Authenticator PAM module. TOTP generates codes that expire after a short time (usually 30 seconds), making it more secure against replay attacks. By comparison, HOTP codes remain valid until used, which can be less secure if a code is intercepted. TOTP is also more widely supported and easier to sync across devices. Typically you'll choose to use TOTP over HOTP, unless you have a specific need for HOTP's counter-based approach.

Conclusion

Google Authenticator is a relatively simple and cost-effective method for configuring your AIX systems for MFA. Of course, if you need a solution that provides additional enhanced security features, such as centralized management, real-time compliance monitoring, real-time auditing, preconfigured security profiles, and multifactor authentication, then a product like IBM's PowerSC is most likely a better fit for your environment. (Refer to the IBM PowerSC training course **[10]**.) However, I encourage you to consider the information discussed here and reflect on how it can further strengthen the security of your AIX systems. ■

Info

[1] PowerSC multifactor authentication concepts: [https://www.ibm.com/docs/en/powersc-mfa/2.3.0?topic=concepts-multi-factor-authentication]

[2] AIX multifactor authentication by RSA SecurID: [https://community.ibm.com/community/user/blogs/xinya-wang1/2020/10/12/aix-multi-factor-authentication-via-rsa-securid]

[3] AIX Toolbox google-authenticator-libpam: [https://community.ibm.com/community/user/discussion/google-authenticator-libpam-is-now-available-on-aix-toolbox]

[4] "Managing FOSS applications on AIX" by Chris Gibson, *ADMIN*, issue 87, 2025, pg. 78: [https://www.admin-magazine.com/Archive/2025/87/Managing-FOSS-applications-on-AIX]

[5] Open source software on IBM AIX: [https://www.ibm.com/training/course/open-source-software-on-ibm-aix-AN35DG]

[6] Enable SSH on AIX to use PAM: [https://www.ibm.com/support/pages/enable-ssh-aix-use-pam]

[7] GitHub google-authenticator-libpam: [https://github.com/google/google-authenticator-libpam]

[8] PAM support in OpenSSH with debugging: [http://emmanuel.iffly.free.fr/doku.php?id=aix:aix_pam_ssh]

[9] GitHub noskewadj for google-authenticator-libpam: [https://github.com/google/google-authenticator-libpam/?tab=readme-ov-file#noskewadj]

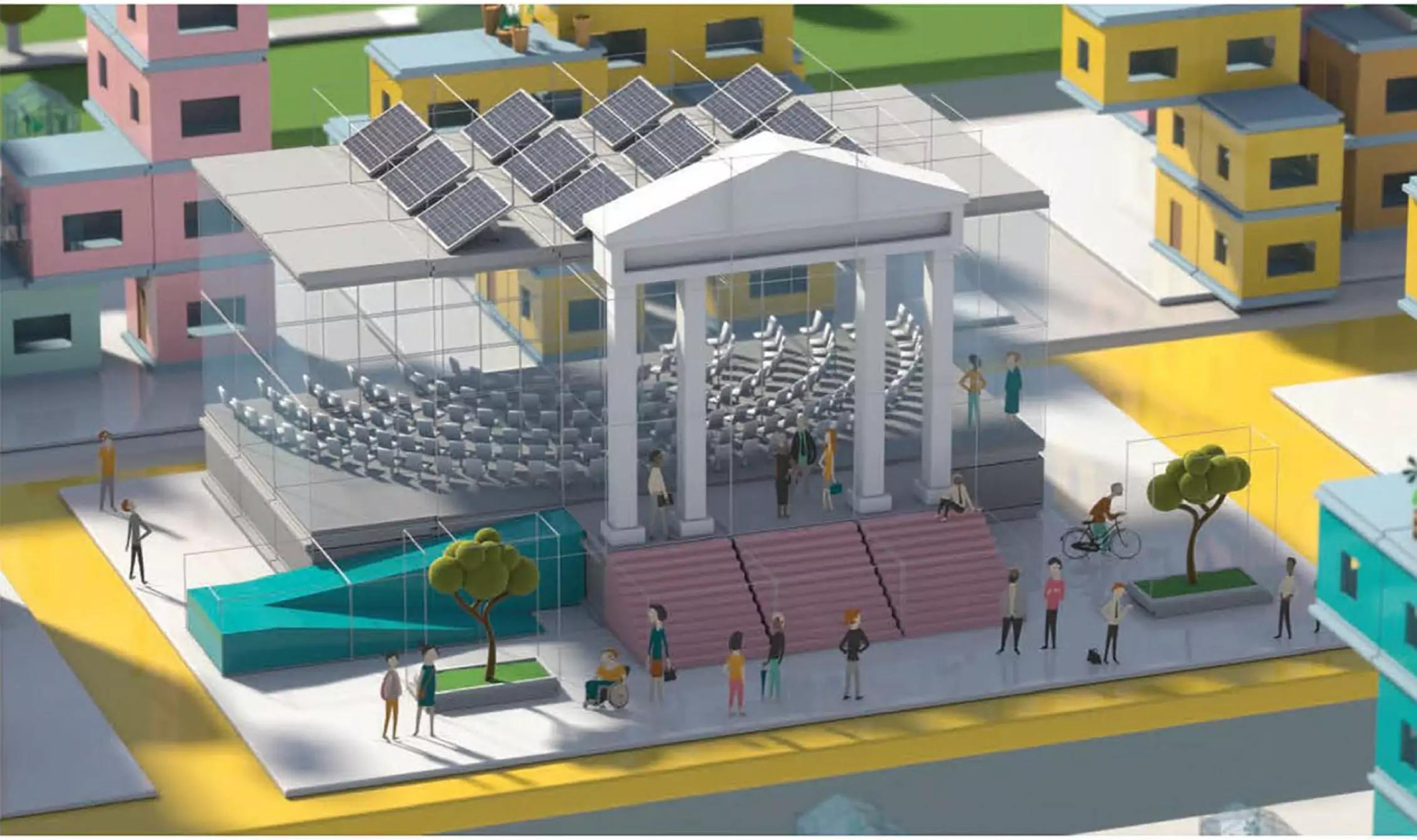
[10] IBM PowerSC workshop: [https://www.ibm.com/training/course/ibm-powersc-workshop-AN80G]

Author

Chris Gibson is a technical specialist on the IBM Power Technical Training team with more than 25 years of experience in AIX, PowerVM, and IBM Power systems. He has authored numerous technical articles, co-authored several IBM Redbooks, and shares insights in his AIX Blog ([ibm.biz/aixblogcg]) and on LinkedIn ([ibm.biz/cg-linkedin]).

■

Public Money Public Code



Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

Photo by Divyansh Jain on Unsplash

Automating Debian workstation enrollment

Elite Enrollment

Puppet enrollment can be automated for enterprise deployment on Debian workstations, but security policy should be enforced with autosign scripts, so the implementation of the policy can be automated with a one-time enrollment script and an automated installation preseed.

By Sam Klein

Imagine it's Friday afternoon: Five new workstations arrive with the urgent requirement that they be ready for orientation on Monday. You're not planning to give up your weekend, and there's no chance you'll come in early Monday. However, each system still needs to be inventoried, labeled, booted, imaged, and configured with every department-required application. That's hours of repetitive work made worse by the fact that the systems are running Linux, not Windows. How do you possibly finish in one afternoon, while tickets continue to stack up?

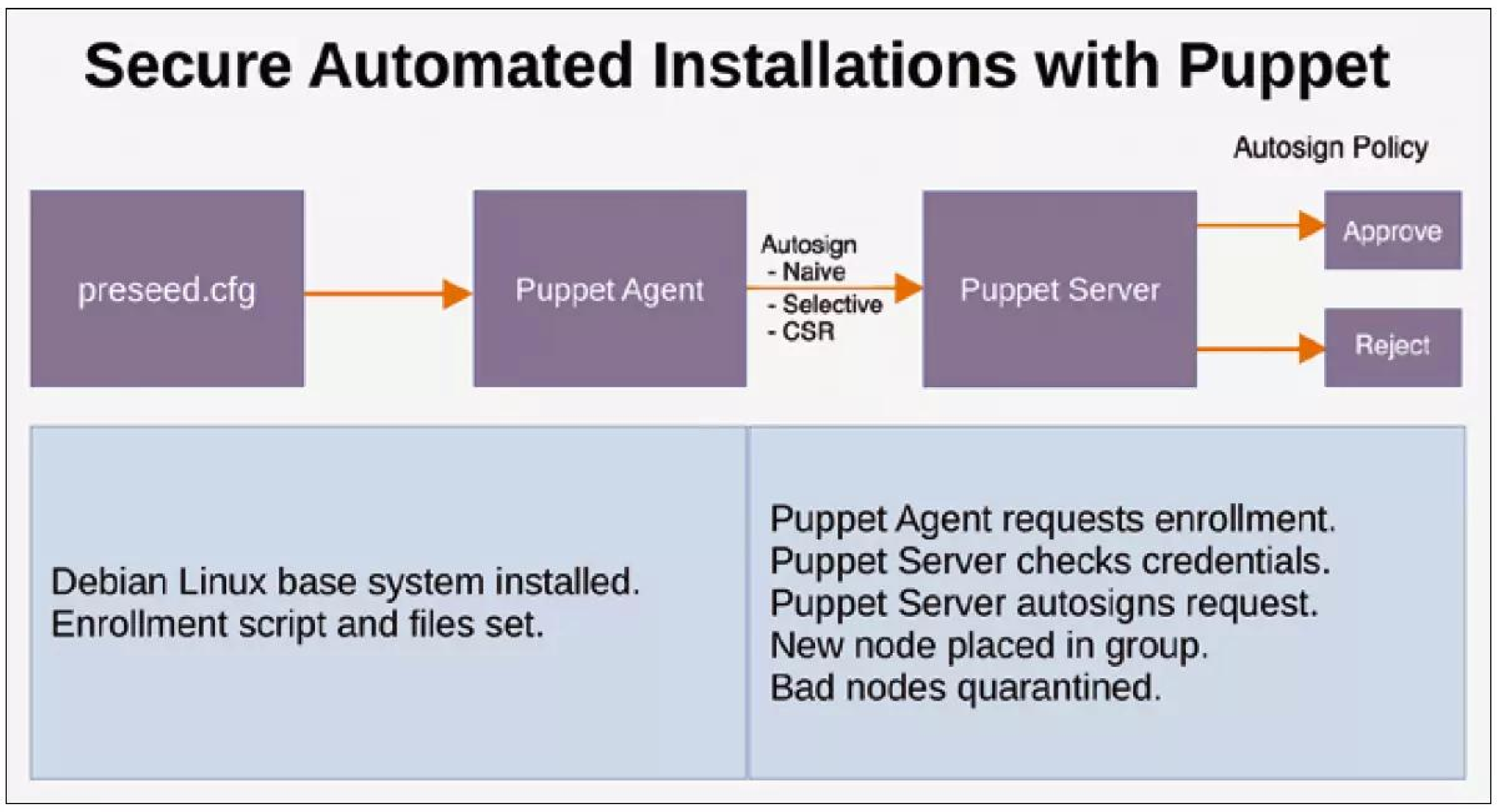
The answer is automated enrollment. By combining Debian preseed installation with Puppet server configuration, you can turn the entire process into a hands-off workflow. All that's required is a USB stick, a bit of preparation, and a reliable network connection (e.g., wired Ethernet). Puppet handles configuration and deployment automatically: no manual domain joins, no one-off tweaks. Systems can be networked, managed, audited, and logged without direct intervention.

Much of this setup requires only minimal configuration, sometimes as little as a single file change on a modern Puppet server. In this tutorial, you'll first see the simplest approach: the use of a wildcard rule to accept all client certificates. From there, you'll strengthen the policy by restricting the wildcard to a few hostnames and checking whether a required file exists as authorization. Finally, you use a method to check custom certificate signing request (CSR) attributes, ensuring that only authorized Linux workstations can enroll (Figure 1). Both approaches remove the need for manual enrollment once the device boots from an approved preseed installer.

The preseed configuration itself provides the necessary authentication files, assigns the device a hostname, defines its department role, and installs requested software according to its category. By the end, you'll be able to deploy fully configured Linux workstations on real hardware without manually configuring each device.

Autosign for Any Puppet Agent

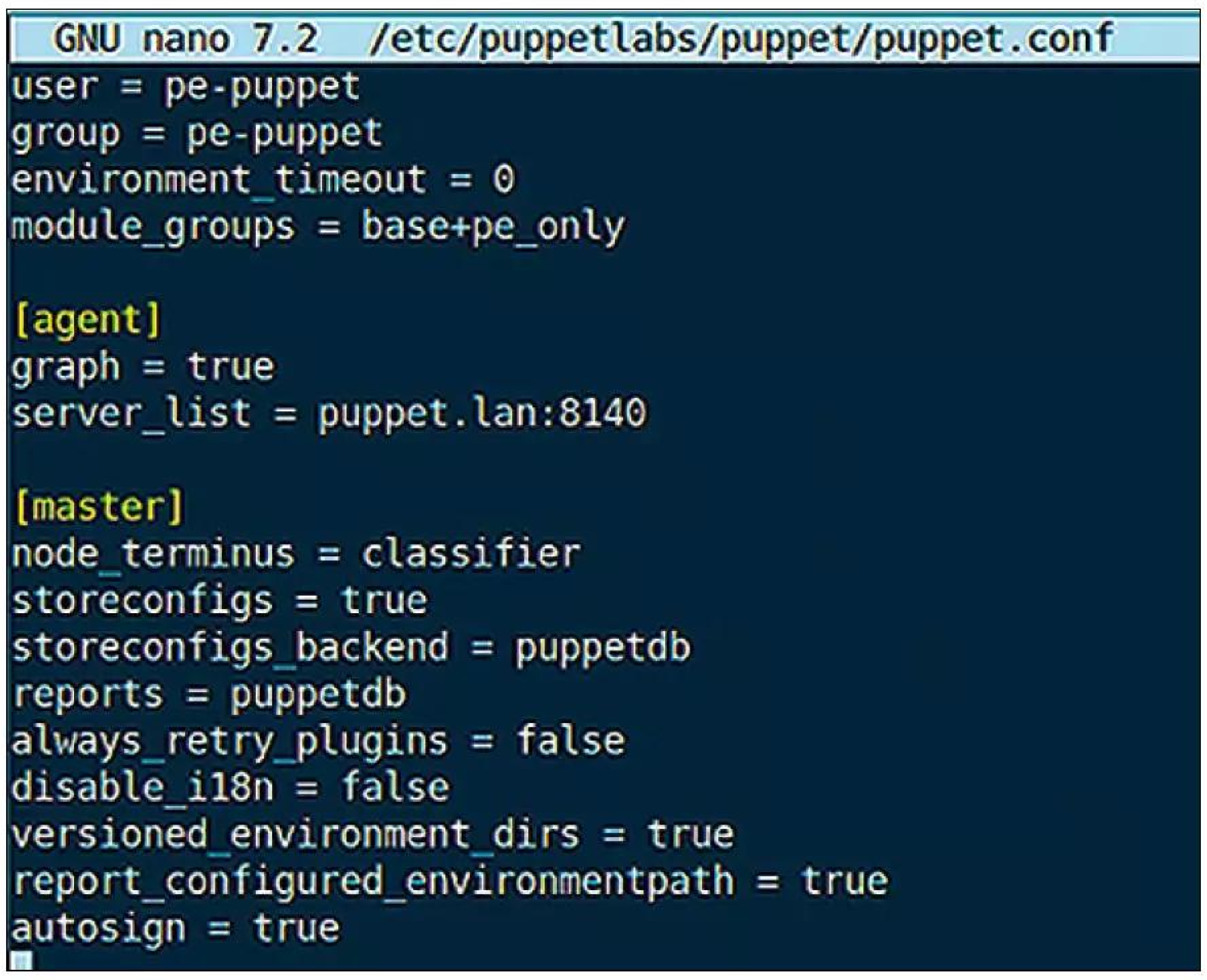
The first approach enables autosign on Puppet Server, allowing any new



Debian Linux base system installed.
Enrollment script and files set.

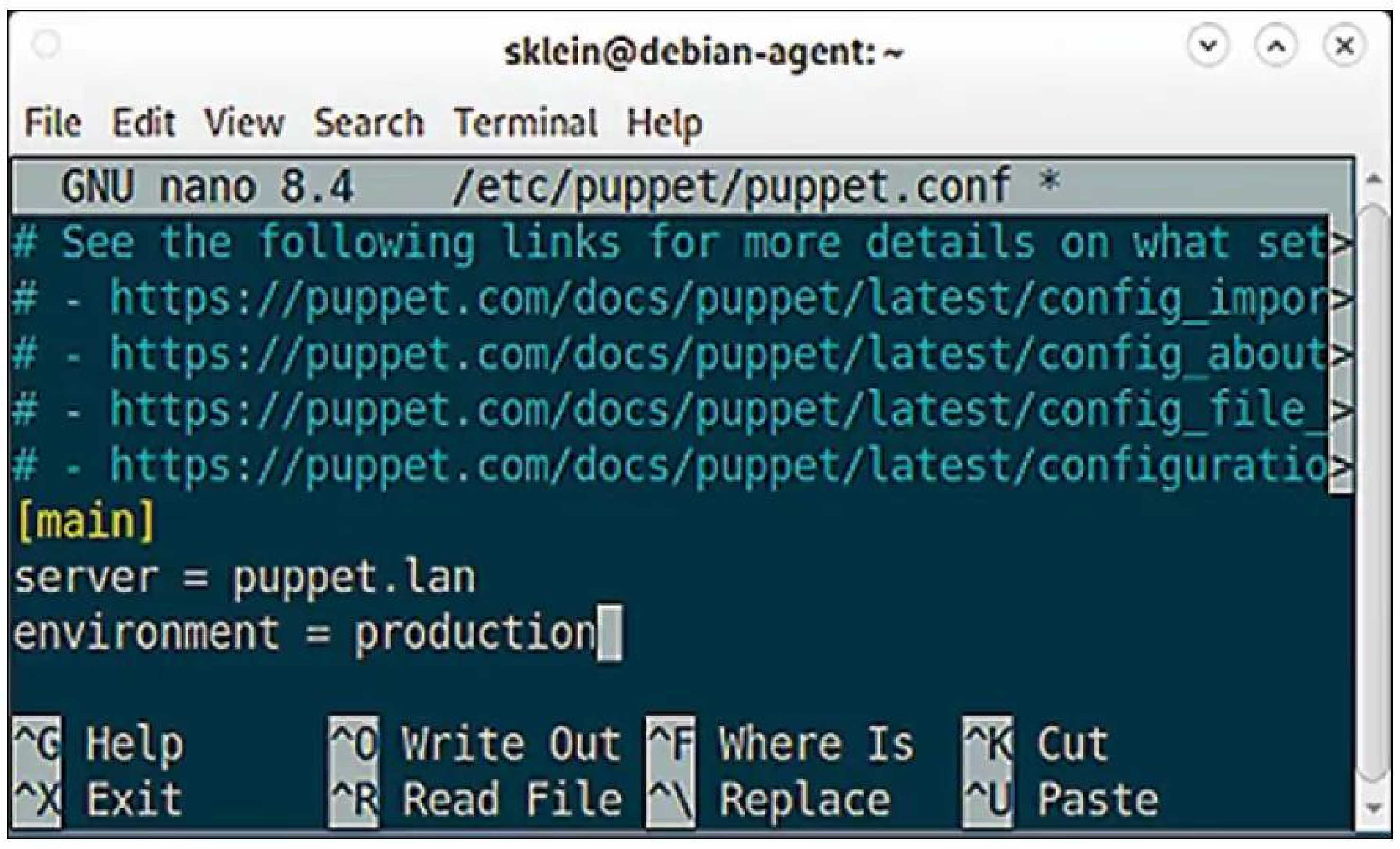
Puppet Agent requests enrollment.
Puppet Server checks credentials.
Puppet Server autosigns request.
New node placed in group.
Bad nodes quarantined.

Figure 1: Automation of workstation deployment begins with the preseed script, which is a Debian-specific method of automation that installs the base system before Puppet enrollment.



```
  GNU nano 7.2   /etc/puppetlabs/puppet/puppet.conf
user = pe-puppet
group = pe-puppet
environment_timeout = 0
module_groups = base+pe_only

[agent]
graph = true
server_list = puppet.lan:8140

[master]
node_terminus = classifier
storeconfigs = true
storeconfigs_backend = puppetdb
reports = puppetdb
always_retry_plugins = false
disable_i18n = false
versioned_environment_dirs = true
report_configured_environmentpath = true
autosign = true
```

Figure 2: At the bottom of the file, add `autosign = true` for a naive autosign policy configuration that accepts all requests.



```
  GNU nano 8.4      /etc/puppet/puppet.conf *
# See the following links for more details on what set>
# - https://puppet.com/docs/puppet/latest/config_impor>
# - https://puppet.com/docs/puppet/latest/config_about>
# - https://puppet.com/docs/puppet/latest/config_file_>
# - https://puppet.com/docs/puppet/latest/configuratio>
[main]
server = puppet.lan
environment = production

^G Help        ^O Write Out ^F Where Is  ^K Cut
^X Exit        ^R Read File ^\ Replace   ^U Paste
```

Figure 3: On the agent device, adding the Puppet address is necessary for enrollment.

agent certificate request to be signed automatically. Puppet's documentation calls this a *naive autosign policy*, and although it is not recommended for production [1], it is a fast way to establish automated enrollment.

Autosign can be set in Puppet Server's `puppet.conf` file under the `[master]` section. The attribute accepts `true`, `false`, or an expression (e.g., a file path). Setting `autosign = true` means all incoming CSRs are automatically approved. A slightly safer option is to point autosign to a policy file,

```
autosign = /<path/to>/autosign.conf
```

which gives some control over which CSRs are accepted. Without autosign enabled, you must manually review and approve every certificate request [2].

To begin, edit the Puppet Server configuration and add either `autosign = true` or the path to the autosign policy file (Figure 2).

Next, configure the `autosign.conf` file. A single asterisk (`*`) in the file as a wildcard with no other characters will permit any hostname (e.g., `desktop.lan`, `workstation.lan`, or `helloworld.lan`). This method is useful for rapid enrollment, although it also expands the potential attack surface.

Autosign speeds up deployment, but naming conventions remain critical for clarity, traceability, and security. Following the guidance in US National Institute of Standards and Technology (NIST) publication SP 800-63A [3], identities should go through a process of validation (ensuring the request meets policy) and verification (confirming authenticity) before a certificate is issued. Autosign should therefore only be used when the CSR has already been validated by other controls. Proper node naming prevents collisions, spoofing, and ambiguity, reinforcing layered security through auditable and accurate identities.

Once autosign is configured, a Puppet agent can begin enrollment. On the workstation, edit the agent's configuration file to point to Puppet Server, and then run:

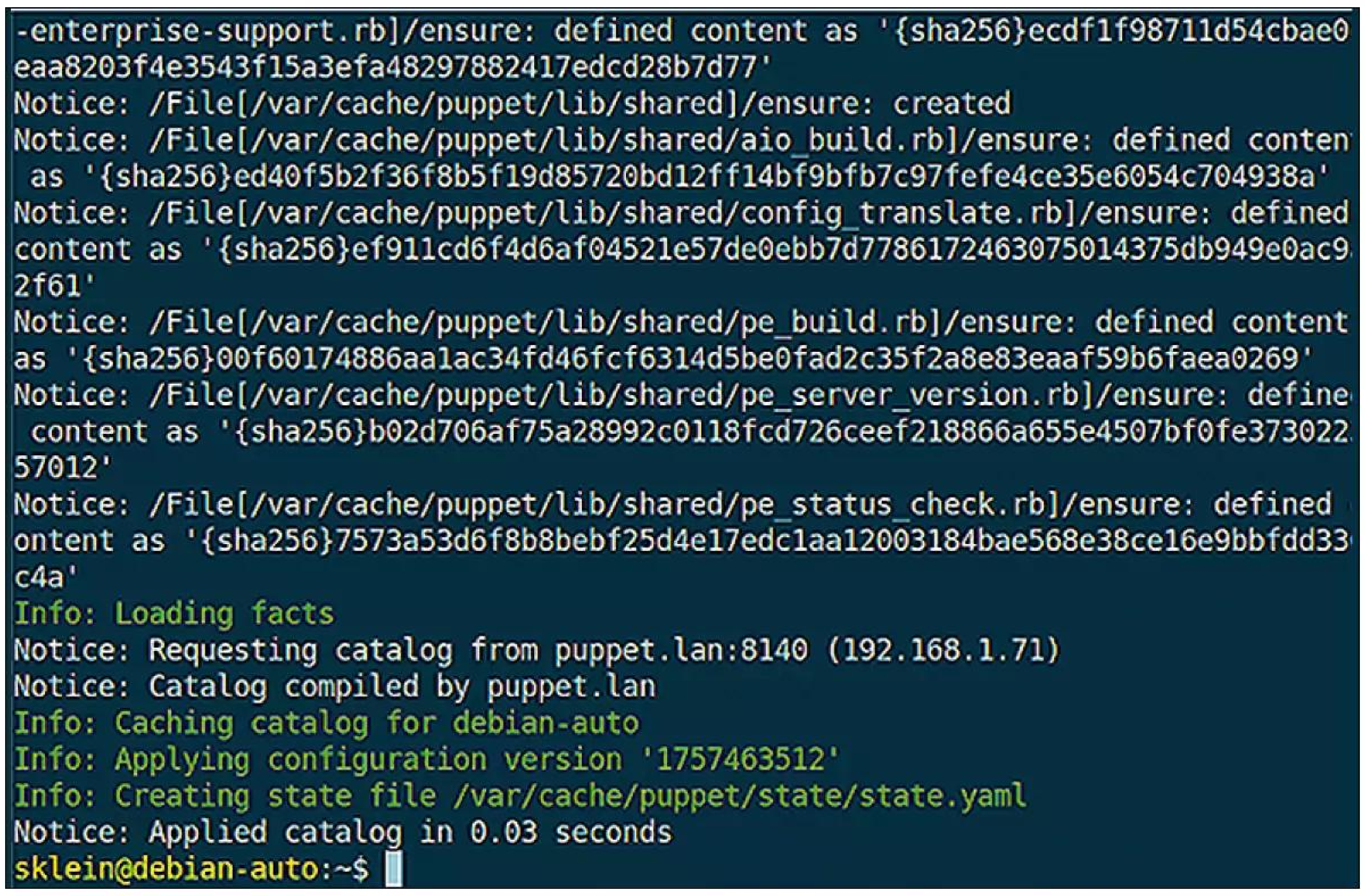
```
puppet agent -t
```

If autosign is active, the agent will be enrolled automatically, and the output log will confirm success.

Before enrollment requests, Puppet agents require that their `puppet.conf` file points at Puppet Server, which can be either an IP or fully qualified domain name (Figure 3).

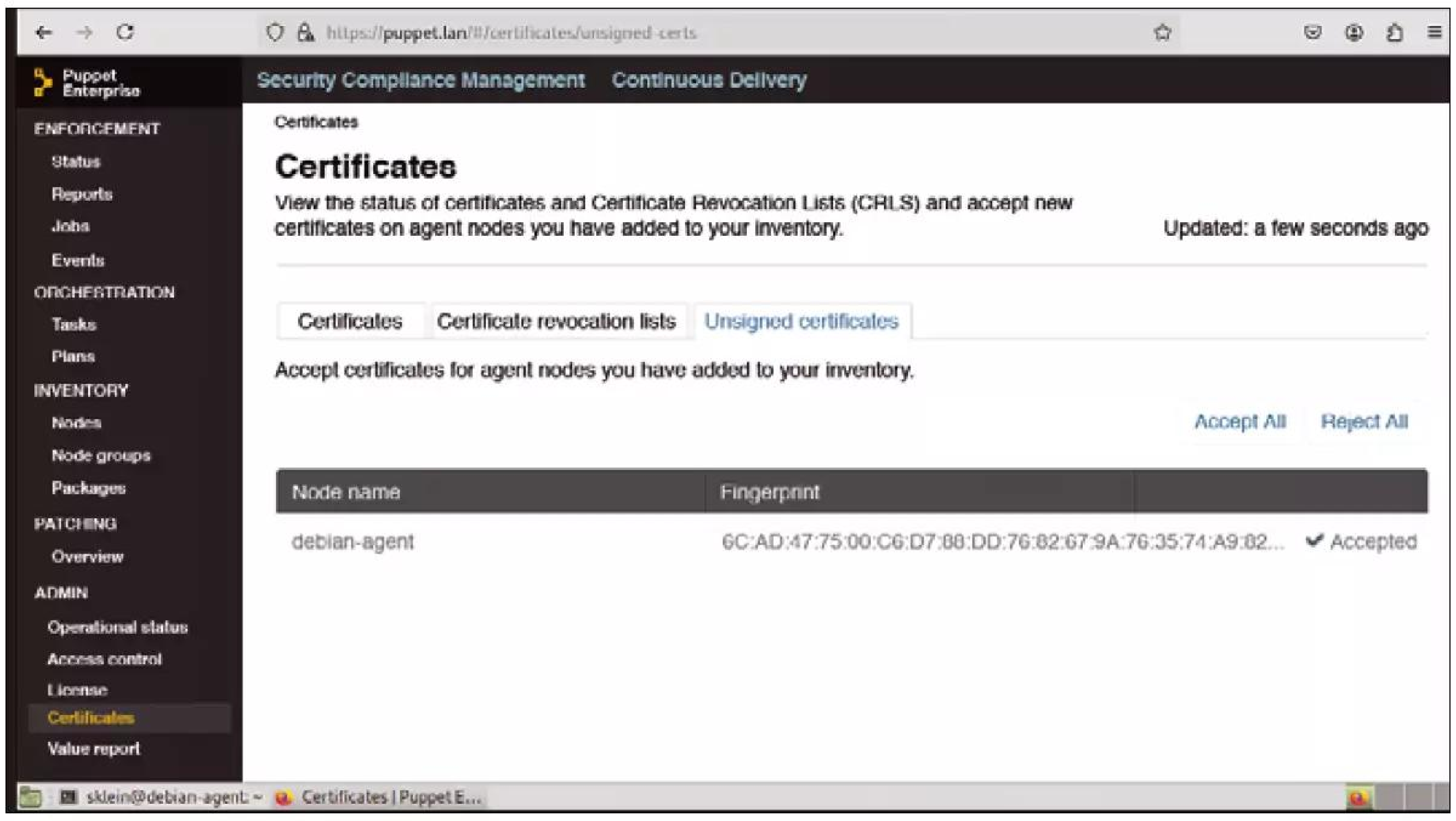
The administrator can verify enrollment through either the Puppet Server command-line interface (Figure 4) or web console (Figure 5). Users running `puppet -t` will see Puppet catalog data as confirmation that the autosign worked (Figure 6).

If autosign is disabled, the CSR remains pending until an administrator approves it. For example, the request would appear, awaiting manual signature (Figure 7).



```
-enterprise-support.rb]/ensure: defined content as '{sha256}ecdf1f98711d54cbae0
eaa8203f4e3543f15a3efa48297882417edcd28b7d77'
Notice: /File[/var/cache/puppet/lib/shared]/ensure: created
Notice: /File[/var/cache/puppet/lib/shared/aio_build.rb]/ensure: defined conten
 as '{sha256}ed40f5b2f36f8b5f19d85720bd12ff14bf9bfb7c97fefe4ce35e6054c704938a'
Notice: /File[/var/cache/puppet/lib/shared/config_translate.rb]/ensure: defined
content as '{sha256}ef911cd6f4d6af04521e57de0ebb7d7786172463075014375db949e0ac9
2f61'
Notice: /File[/var/cache/puppet/lib/shared/pe_build.rb]/ensure: defined content
as '{sha256}00f60174886aa1ac34fd46fcf6314d5be0fad2c35f2a8e83eaaf59b6faea0269'
Notice: /File[/var/cache/puppet/lib/shared/pe_server_version.rb]/ensure: define
 content as '{sha256}b02d706af75a28992c0118fcd726ceef218866a655e4507bf0fe373022
57012'
Notice: /File[/var/cache/puppet/lib/shared/pe_status_check.rb]/ensure: defined
ontent as '{sha256}7573a53d6f8b8bebf25d4e17edc1aa12003184bae568e38ce16e9bbfdd33
c4a'
Info: Loading facts
Notice: Requesting catalog from puppet.lan:8140 (192.168.1.71)
Notice: Catalog compiled by puppet.lan
Info: Caching catalog for debian-auto
Info: Applying configuration version '1757463512'
Info: Creating state file /var/cache/puppet/state/state.yaml
Notice: Applied catalog in 0.03 seconds
sklein@debian-auto:~$
```

Figure 4: Once autosign has been enabled, the Puppet agent will receive the catalog at the time it makes the request.



Certificates

Certificates

View the status of certificates and Certificate Revocation Lists (CRLS) and accept new certificates on agent nodes you have added to your inventory.

Updated: a few seconds ago

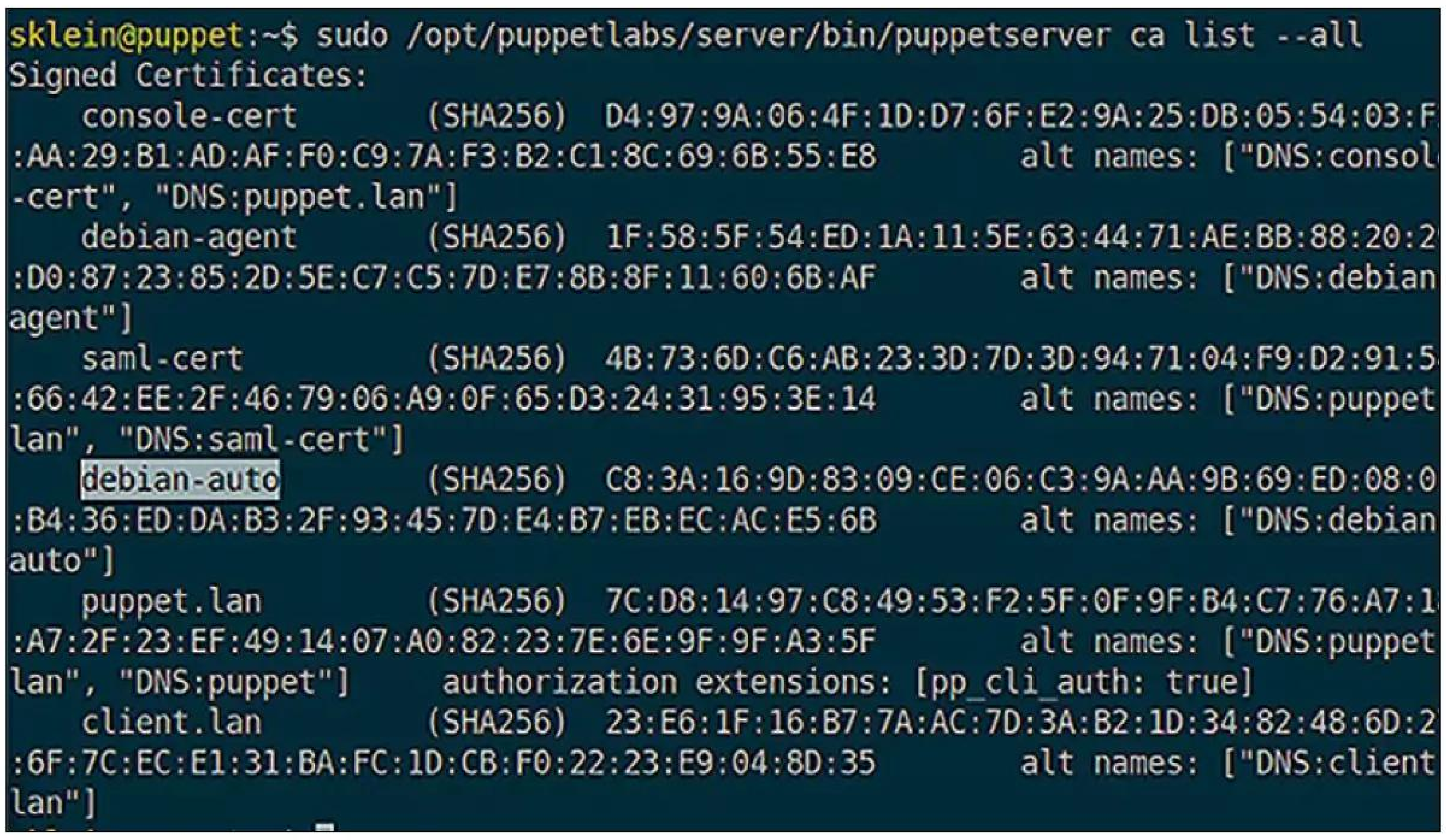
Certificates | Certificate revocation lists | Unsigned certificates

Accept certificates for agent nodes you have added to your inventory.

Accept All Reject All

Node name	Fingerprint	
debian-agent	6C:AD:47:75:00:C6:D7:88:DD:76:82:67:9A:76:35:74:A9:82...	✔ Accepted

Figure 5: Administrators can use the web portal to sign a request manually.



```
sklein@puppet:~$ sudo /opt/puppetlabs/server/bin/puppetserver ca list --all
Signed Certificates:
    console-cert        (SHA256)  D4:97:9A:06:4F:1D:D7:6F:E2:9A:25:DB:05:54:03:F
:AA:29:B1:AD:AF:F0:C9:7A:F3:B2:C1:8C:69:6B:55:E8        alt names: ["DNS:consol
-cert", "DNS:puppet.lan"]
    debian-agent        (SHA256)  1F:58:5F:54:ED:1A:11:5E:63:44:71:AE:BB:88:20:2
:D0:87:23:85:2D:5E:C7:C5:7D:E7:8B:8F:11:60:6B:AF        alt names: ["DNS:debian
agent"]
    saml-cert           (SHA256)  4B:73:6D:C6:AB:23:3D:7D:3D:94:71:04:F9:D2:91:5
:66:42:EE:2F:46:79:06:A9:0F:65:D3:24:31:95:3E:14        alt names: ["DNS:puppet
lan", "DNS:saml-cert"]
    debian-auto         (SHA256)  C8:3A:16:9D:83:09:CE:06:C3:9A:AA:9B:69:ED:08:0
:B4:36:ED:DA:B3:2F:93:45:7D:E4:B7:EB:EC:AC:E5:6B        alt names: ["DNS:debian
auto"]
    puppet.lan          (SHA256)  7C:D8:14:97:C8:49:53:F2:5F:0F:9F:B4:C7:76:A7:1
:A7:2F:23:EF:49:14:07:A0:82:23:7E:6E:9F:9F:A3:5F        alt names: ["DNS:puppet
lan", "DNS:puppet"]     authorization extensions: [pp_cli_auth: true]
    client.lan          (SHA256)  23:E6:1F:16:B7:7A:AC:7D:3A:B2:1D:34:82:48:6D:2
:6F:7C:EC:E1:31:BA:FC:1D:CB:F0:22:23:E9:04:8D:35        alt names: ["DNS:client
lan"]
```

Figure 6: Without intervention, the agent's request was signed, as found with the `puppetserver ca list --all` command in a terminal client.

Selective Autosign Policy

A second and more secure alternative to the naive autosign policy is the *selective autosign policy*. Instead of allowing all certificates, the `autosign.conf` file specifies exactly which hostnames may be signed automatically. Although this method improves on the "accept anything" model, it still has a critical weakness: An attacker might guess or spoof an acceptable hostname, which is considered weak authentication because the client provides only an identifier (its hostname) without actual proof of authenticity [4].

Listing 1: Selective Autosign Policy

```
#!/bin/bash

FACTFILE="/etc/puppet-passphrase"

if [ -f "$FACTFILE" ]; then
    if grep -qx "secret" "$FACTFILE"; then
        echo "auth_check=valid"
    else
        echo "auth_check=invalid"
    fi
else
    echo "auth_check=missing"
fi
```

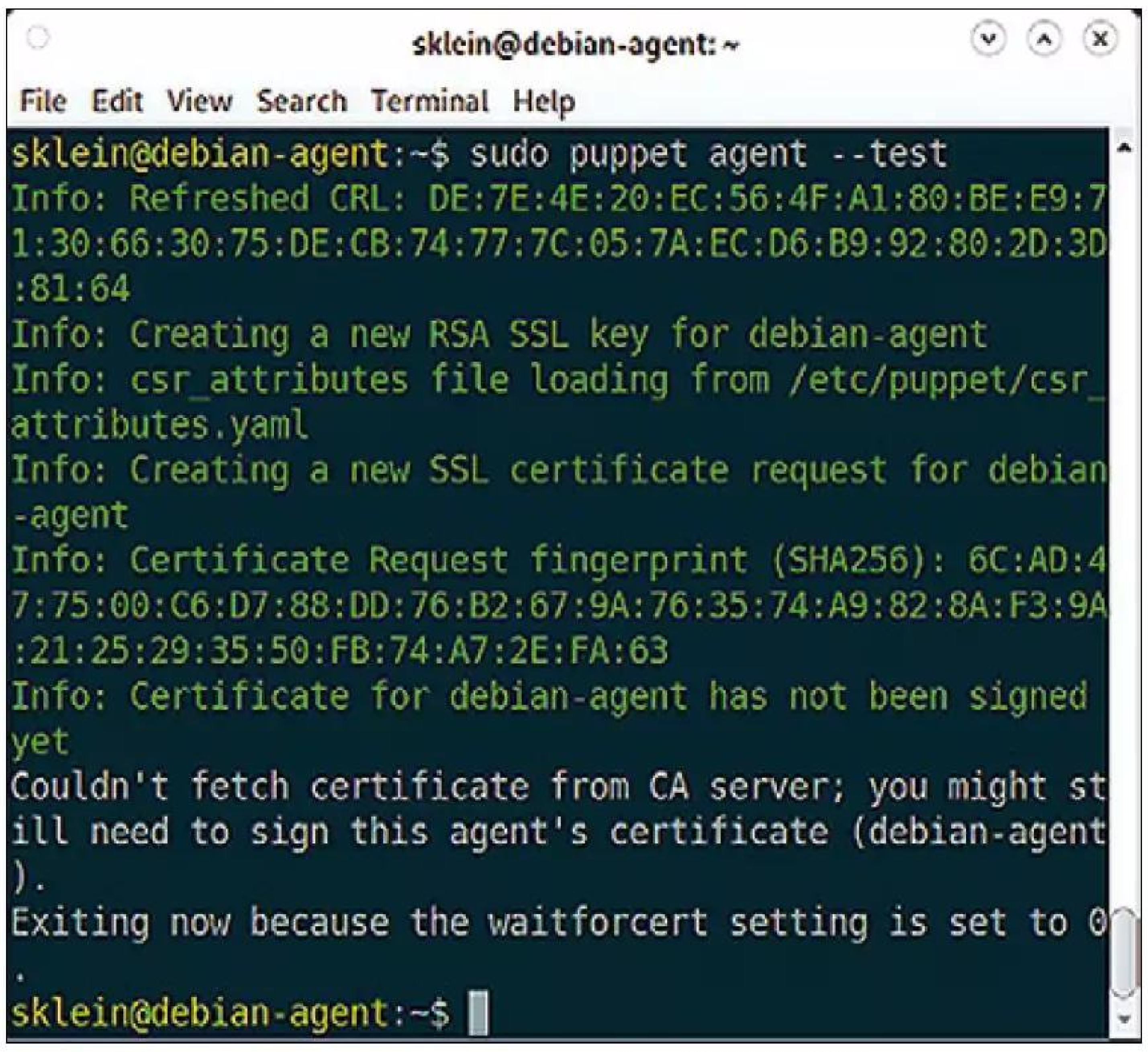
To address this weakness, Puppet can enforce an additional fact-based check after enrollment. By attaching a fact to each node, Puppet categorizes new devices with evidence beyond hostname alone. Unauthorized or suspicious nodes can be automatically placed in a quarantine group for administrator review, while legitimate systems are sorted into the proper production category.

Facts in Puppet are attributes collected by the Facter library [5][6], which profiles system details such as operating system, filesystem, and hardware. You can extend facts with external facts defined by custom scripts. The Bash script in Listing 1 defines a fact called `auth_check` that looks for a file named `/etc/puppet-passphrase` and checks whether it contains the correct secret string (Figure 8).

If you save this script as `/etc/puppetlabs/facter/facts.d/authcheck.sh`, then once the Puppet agent syncs (e.g., with `puppet agent --test`), the fact is visible in Puppet as `auth_check`.

Next, configure Puppet to sort nodes into categories according to the fact. From the Puppet Server console, go to *Nodes* | *Classification* | *Add group*. Place the new group under *production* (or the appropriate Environment). The group will appear under *Node groups* (Figure 9).

In this example, I create a group named *Quarantine*. Any signed node is considered, but only those with a



```
sklein@debian-agent:~$ sudo puppet agent --test
Info: Refreshed CRL: DE:7E:4E:20:EC:56:4F:A1:80:BE:E9:7
1:30:66:30:75:DE:CB:74:77:7C:05:7A:EC:D6:B9:92:80:2D:3D
:81:64
Info: Creating a new RSA SSL key for debian-agent
Info: csr_attributes file loading from /etc/puppet/csr_
attributes.yaml
Info: Creating a new SSL certificate request for debian
-agent
Info: Certificate Request fingerprint (SHA256): 6C:AD:4
7:75:00:C6:D7:88:DD:76:B2:67:9A:76:35:74:A9:82:8A:F3:9A
:21:25:29:35:50:FB:74:A7:2E:FA:63
Info: Certificate for debian-agent has not been signed
yet
Couldn't fetch certificate from CA server; you might st
ill need to sign this agent's certificate (debian-agent
).
Exiting now because the waitforcert setting is set to 0
.
sklein@debian-agent:~$ 
```

Figure 7: When an autosign policy fails or rejects a request from an agent, the client will receive a notice that signing the request is required by an administrator.

valid `auth_check` fact are allowed into production. When Puppet evaluates a node, it checks the fact value. Nodes without the proper `/etc/puppet-passphrase` file, or with an incorrect secret, are automatically moved into *Quarantine*. In this example, `puppet.lan` is valid (as set earlier; see Figure 3), while an unauthorized node would be isolated.

By combining selective autosign with fact logic, Puppet maintains automation while reducing the risk of spoofed hostnames. This layered approach ensures that autosign does not bypass essential identity checks, providing both efficiency and policy compliance.

Autosign with Custom CSR Attributes

The third method builds on the previous approaches by adding a CSR attribute as proof of identity. Puppet supports embedding a pre-shared key (PSK) into the CSR with the `challengePassword` attribute (OID policy 1.2.840.113549.1.9.7), as recommended in the Puppet documentation [7].

When an agent generates its CSR, the `challengePassword` is included. Puppet Server then evaluates the CSR against a policy-driven autosign script, which extracts the `challengePassword` value, verifies it against the expected secret, and either approves or rejects the request. Because CSRs are transmitted over TLS, the secret is encrypted in transit. This method can be combined with fact-based checks (e.g., `auth_check`) and group logic (e.g., Quarantine) to form a layered authentication policy, potentially extended with multifactor authentication (MFA) or hardware tokens.

On each Puppet agent, you define a CSR attribute file at `/etc/puppet/csr_attributes.yaml`:

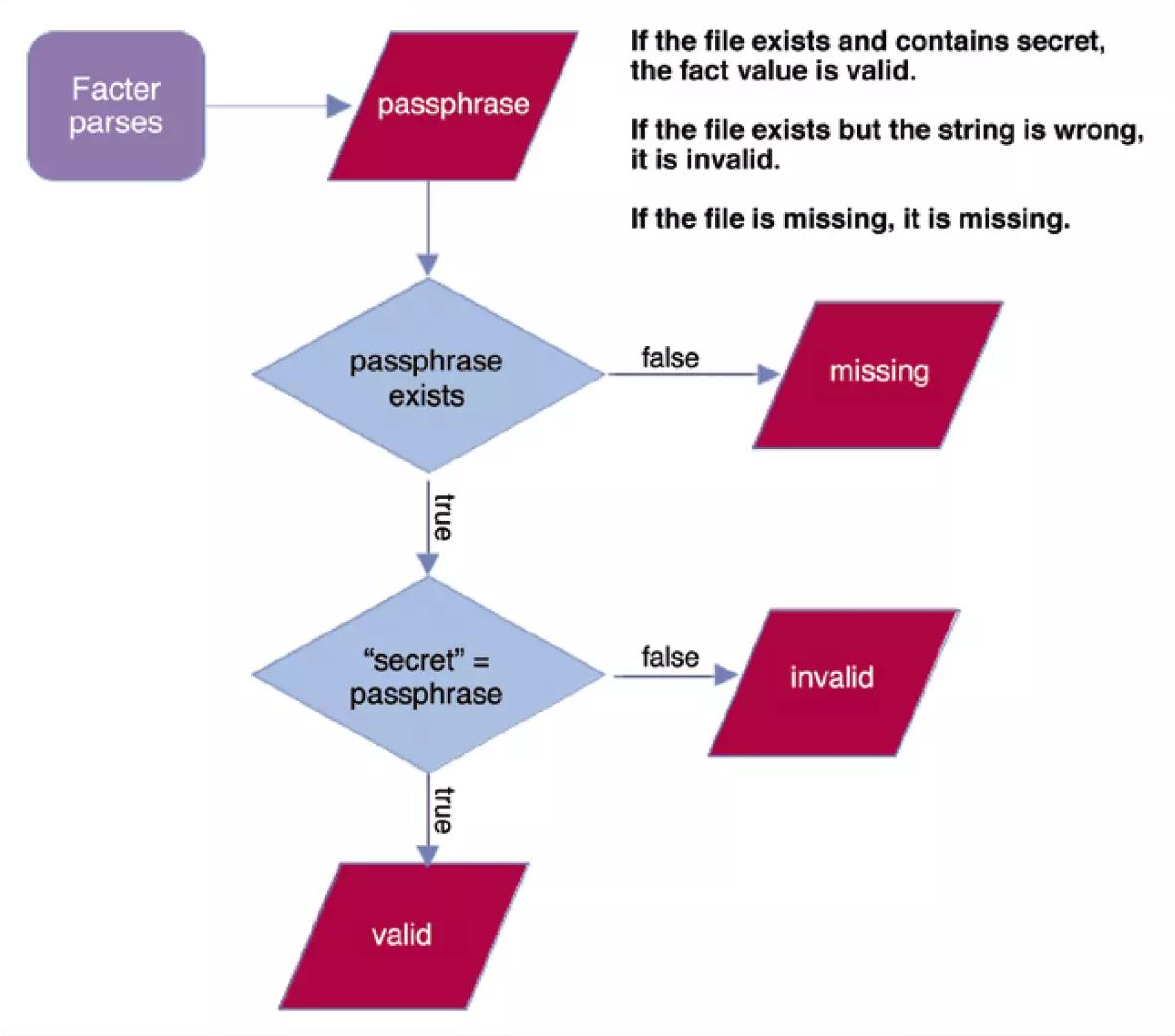
```
custom_attributes:
  challengePassword: Secret
```

This file ensures the `challengePassword` field is included whenever the agent generates its CSR.

On Puppet Server, you configure autosign in `puppet.conf` to point to a script. The script (Listing 2) reads the CSR from the `requests` directory, extracts the `challengePassword`, and validates it [8].

Listing 2: Parsing the Password from CSR

```
#!/bin/bash

PASSWORD="Secret"
CERT_NAME=${1:-}

# Validate certificate name input
if [[ -z "$CERT_NAME" ]] || [[ ! "$CERT_NAME" =~
   ^[a-zA-Z0-9._-]+$ ]]; then
    exit 1
fi

REQUEST_DIR="/etc/puppetlabs/puppetserver/ca/requests"
CERT_FILE="$REQUEST_DIR/$CERT_NAME.pem"

# Ensure CSR file exists and is readable
if [[ ! -f "$CERT_FILE" ]] || [[ ! -r "$CERT_FILE" ]]; then
    exit 1
fi

# Extract challengePassword from CSR
agent_password=$(openssl req -noout -text -in "$CERT_
   FILE" 2>/dev/null | awk -F ':' '/challengePassword/
   {gsub(/^[ \t]+|[ \t]+$/, "", $2); print $2; exit}'
   || true)

# Validate challengePassword
if [[ -z "$agent_password" ]]; then
    exit 1
fi

if [[ "$agent_password" == "$PASSWORD" ]]; then
    exit 0  # approve
else
    exit 1  # reject
fi
```

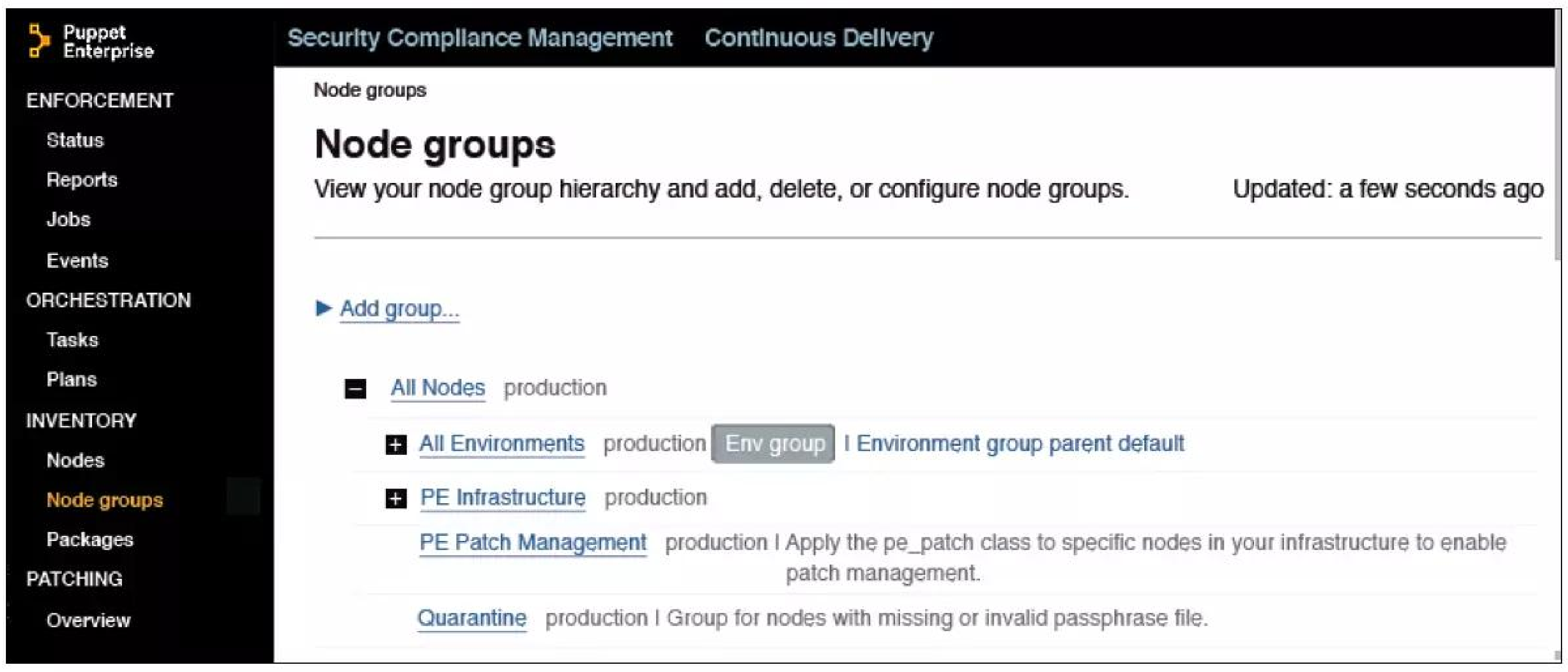


If the file exists and contains secret, the fact value is valid.

If the file exists but the string is wrong, it is invalid.

If the file is missing, it is missing.

Figure 8: If the file exists and contains `secret`, the fact value is valid. If the file exists but the string is wrong, it is invalid. If the file is missing, it is missing.



Node groups

Node groups

View your node group hierarchy and add, delete, or configure node groups. Updated: a few seconds ago

All Nodes production

All Environments production Env group I Environment group parent default

PE Infrastructure production

PE Patch Management production I Apply the pe_patch class to specific nodes in your infrastructure to enable patch management.

Quarantine production I Group for nodes with missing or invalid passphrase file.

Figure 9: Puppet sorts nodes into categories (node groups).

This script includes multiple safeguards. Input validation ensures `CERT_NAME` is not empty and contains only safe characters. File checks confirm that the CSR file exists and is readable. Sanitization prevents attacks such as path traversal [9], command injection [10], shell globbing [11], or local file inclusion (LFI) [12]. Controlled exit codes (`0` to approve, `1` to reject) make the policy explicit and reliable.

The autosign policy script validates each incoming CSR before Puppet Server decides whether to approve it. The goal is to prevent unauthorized enrollment by checking for a shared secret embedded in the CSR's `challengePassword` attribute.

The `PASSWORD="Secret"` line sets the trusted passphrase expected in the CSR's `challengePassword` field. In production, this should be rotated, hashed, or replaced with a stronger authentication mechanism.

The `CERT_NAME=${1:-}` line assigns the first positional argument passed to the script. When Puppet invokes an autosign script, it calls it with the certificate name (e.g., `workstation1.lan`) as `$1`, which is the standard input (stdin), not as an argument. The `${1:-}` syntax means "use `$1` if it's provided; otherwise, fall back to an empty string."

The script checks that the certificate name is not empty and contains only safe characters (letters, digits, dots, underscores, and hyphens). This prevents malicious inputs such as path traversal (`../../`) or command injection attempts. The script then uses `openssl` to extract the CSR's `challengePassword` field. The `awk` trims whitespace and outputs only the value. If no field is found, the variable is left empty. Finally, the extracted password is compared with the trusted secret. If they match, the script exits with `0`, signaling Puppet to approve the CSR automatically. If they don't match (or if the value is missing), the script exits with `1`, rejecting the request.

After writing the script, make it executable and ensure ownership belongs to the Puppet user (adjust as needed for your environment, e.g., `pe-puppet` for Enterprise):

```
chmod +x /etc/puppetlabs/puppet/script.sh
chown pe-puppet ➥
  /etc/puppetlabs/puppet/script.sh
```

If autosign does not appear to work, confirm that no `autosign.conf` file exists in `/etc/puppetlabs/puppet/`, because it will override the script-based policy. A server reboot might also help refresh the policy in some environments.

Additional Steps for Security

CSR authentication enables a robust security policy without accepting all requests or writing out each device into a file as an allow list (also called whitelist). However, the current demonstration is not enough. PSK authentication is weak; anyone can guess the password and have liberal permissions without MFA or other checks. Mitigating this risk should be done with a few recommendations, tiered by difficulty of implementation: PSK rotation and hashing.

PSK rotation is perhaps the easiest because it requires that you switch the secret routinely but needs no encryption knowledge; however, it does not mitigate against an attacker retrieving the key, so during the period before the switch, an attack can occur. This method is the easiest to implement because no scripting or automation is necessary besides file changes.

One method to obscure the secret then is to hash it (**Listing 3**). Hashing can be done with a script in which the user conceals the string with what appears to be random characters. Hashes typically aren't used as a way to conceal and reveal information; they are more effective at validating files. In this setup, the hostname, something both you and the local host knows, is used to conceal the secret that is unknown to the user but is known by you. Therefore you can reveal what the hash says by reconstructing the hash and matching it. The result is a string of characters that the user cannot read, but you can verify, such as this hash in the `csr_attributes.yaml` file:

```
custom_attributes:
  challengePassword: ➥
   e25f289c4e3734b0c1fc8604564a7ebd2b ...
```

To verify `challengePassword`, you simply pass the same string as an echo to the same hashing algorithm (e.g., SHA256) from an admin's terminal:

```
echo -n "$SECRET$HOSTNAME" | sha256sum
```

Furthermore, encryption keys can be used to encrypt the passphrase that then can be interpreted by both you and Puppet. This method would require more protections on the installer itself and the keys designated. Therefore, it might be more trouble to maintain encryption keys for the installer rather than hashing and a secret rotation. Additional input of the hash can further secure the string from tampering and enhance the confidentiality.

By layering hostname checks, fact-based logic (e.g., `auth_check`), and CSR attributes, Puppet can enforce a robust authentication chain without manual intervention, reducing the risk of spoofed or unauthorized nodes entering production while still preserving the efficiency of automated enrollment.

Automation with Preseeds

Unlike the leap from selective autosign policy to a more refined policy with CSR, the use of a preseed

Listing 3: Hashing the Secret

```
#!/bin/bash
SECRET="Secret"
HOSTNAME=$(hostname)

HASHED_SECRET=$(echo -n "${SECRET}${HOSTNAME}" | sha256sum | awk '{print $1}')

cat > /etc/puppet/csr_attributes.yaml <<EOF
custom_attributes:
  challengePassword: $HASHED_SECRET
EOF
```

Listing 4: Preseed Example

```
d-i preseed/late_command string in-target wget -O /usr/local/bin/enroll.sh http://example.lan/~sklein/debian/enroll.sh; in-target chmod +x /usr/local/
    bin/enroll.sh; in-target wget -O /etc/xdg/autostart/enroll.desktop http://example.lan/~sklein/debian/enroll.desktop
```

requires some specialized knowledge. This portion of the process takes a considerable amount of time, yet a number of tools are at hand, and not all require any significant effort. You might not need to automate something the same way – or as completely; therefore, it's easier to develop. In this section I look at one demonstration while suggesting other tools. With a script I've hacked together, I create an ISO image of the Debian installer, run the automated installer, and then wait as the Puppet enrollment script runs.

The first step requires writing a `preseed.txt` file; you can grab an existing example from the Debian documentation [13]. The key to the installer is to create one that does the partitioning, networking, default users, and maybe the name. In some cases I like to enter the name as the only input. You could also have your script name the workstation according to some business logic.

I also chose not to use a desktop environment, so I could reduce the build time and allow myself to select my desktop experience in the post-install script. Another important step to take with the preseed is to create files in the `late` command section (Listing 4).

The `d-i` keyword signifies a Debian installer directive. At the `preseed/late_command` installer variable, the installer will run custom scripts, which I condensed as simple download and file manipulation commands rather than risk a failed installation with a more complicated command failing at runtime. Checking the `d-i` commands you write can be done on a Debian system with the command

```
debconf-set-selections -c preseed.cfg
```

for validation [14].

Finding the commands to use can be accomplished by inspecting a Debian installation log or running an existing preseed file. The Debian installation guide also provides some examples like localization [15]:

```
d-i debian-installer/➥
  locale string en_US.UTF-8
d-i console-setup/ask_detect boolean false
d-i keyboard-configuration/➥
  xkb-keymap select us
```

Similar to `late_command`, the command follows the pattern:

```
d-i [keyword]/[keyword] [class type] ➥
    [value]
```

Some options allow for `auto`, as well. For example, networking can be done without prompting the user if connected with an enabled and available wired network connection. Also by commenting out the command (or deleting the line from the preseed file), the installer will prompt the user for the value:

```
d-i netcfg/choose_interface select auto
# d-i netcfg/get_hostname string debian
d-i netcfg/get_domain string local
```

In the networking example, `debian` was the preseed value for the hostname, but with it commented out, the installer will ask the user to provide a hostname value, which can be useful if you want to create unique hostnames before enrolling with Puppet – a requirement for the CSR and autosign policy.

Software mirrors, accounts, clock, partitioning, and package selection are created similarly with `tasksel`; however, the command differs a bit in syntax and is used primarily for the desktop choice (e.g., Gnome):

```
tasksel tasksel/first multiselect ➥
  standard, gnome-desktop
```

A nearly complete preseed example with many options commented out (which you can enable by removing the `#` symbol at the beginning of the line) is available on the Debian documentation website [16].

Notice in the example that I wrote two files: `enroll.sh` and `enroll.desktop` (Listings 5 and 6). Both are required to start up `enroll.sh` on login. The `enroll.sh` script installs `puppet-agent` and the files needed with CSR custom attributes and then sends the request to Puppet Server.

You can execute the script in various ways. I've found XDG works in my process, which uses a desktop file inside the system `xdg` directory (e.g., `/etc/xdg/autostart`).

Another way is to place the `enroll` script into a dot file like `.profile`, which will require you to log in; it might be beneficial depending on the content of the `enroll` script. However, once you have Puppet enrolled, you might as well do all the installation steps through Puppet instead of with individual scripts. In my production applications, a system configuration tool deletes the default user, hostname, and enrollment files, including any PSKs or confidential files, although it's best practice not to have any confidential files at all and might be beneficial to use encryption keys rather than a PSK.

Preseeds can be either used on the local media by attaching it to an ISO, used remotely from the installer menu, or put into a PXE pipeline.

You can use an ISO creation tool like Cubic [17] or make your own [18],

Listing 5: Enrollment Script to Run on Local Host

```
#!/bin/bash
apt install puppet-agent -y
touch /etc/puppet/csr_attributes.yaml
echo "custom_attributes:" > /etc/puppet/csr_attributes.yaml
echo "  challengePassword: Secret" >> /etc/puppet/csr_attributes.yaml
echo "[main]" > /etc/puppet/puppet.conf
echo "server = puppet.lan" >> /etc/puppet/puppet.conf
puppet agent -t
```

appending the preseed to the ISO and ensuring that an automatic installation menu entry is available. You can also skip creating an ISO at the cost of having to put in the preseed location in the installer menu. This option is found in the installer's *Advanced options* | *Automated install* menu entry; then, write the URL where the preseed is located once networking has been configured.

Once you select *Automated install*, a window explaining the different `preseed.cfg` load options are made apparent (Figure 10). Submit the file location in the input text field.

Conclusion

Automating Linux workstation enrollment is not about finding a single perfect method, it's about balancing efficiency, security, and maintainability. Naive autosign delivers speed but sacrifices assurance and integrity. Selective autosign with fact logic adds accountability and categorization while still allowing automation. CSR attributes with policy scripts raise the bar to strong authentication, aligning with good security practices and compliance frameworks such as NIST SP 800-63A.

The important point is not whether you adopt every layer described here, but that you build automation with intention. Treat autosign as part of a broader identity lifecycle: validation, verification, and issuance. Make naming conventions traceable, enforce policies that are auditable, and design your preseeds so that no secrets linger on disk longer than absolutely necessary.

By approaching enrollment this way, you turn what once demanded hours of repetitive configuration into a streamlined, policy-driven workflow. The Friday scenario of five new workstations needing to be ready by Monday no longer feels like a crisis. Instead of late nights or weekend labor, you boot the machines, let the preseeds run, and watch Puppet handle the rest. ■

Listing 6: Desktop File Example

```
[Desktop Entry]
Type=Application
Name=Enroll
Exec=/usr/local/bin/enroll.sh
```

Info

[1] Autosigning certificate requests: [https://www.puppet.com/docs/puppet/7/ssl_autosign.html]

[2] CSR attributes and certificate extensions: [https://www.puppet.com/docs/puppet/7/ssl_attributes_extensions]

[3] NIST SP 800-63A-4: [https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63A-4.pdf]

[4] Weak authentication: [https://cwe.mitre.org/data/definitions/1390.html]

[5] External facts: [https://www.puppet.com/docs/puppet/7/external_facts#external_facts]

[6] Facter: [https://www.puppet.com/docs/puppet/7/facter]

[7] Certificate extensions: [https://www.puppet.com/docs/puppet/7/config_file_csr_attributes.html]

[8] Configuring Puppet's policy-based autosigning: [https://www.youtube.com/watch?v=vPljzRZ_Ug8]

[9] Path traversal: [https://owasp.org/www-community/attacks/Path_Traversal]

[10] Command injection: [https://owasp.org/www-community/attacks/Command_Injection]

[11] Security implications of forgetting to quote a variable in Bash/POSIX shells: [https://unix.stackexchange.com/questions/171346/security-implications-of-forgetting-to-quote-a-variable-in-bash-posix-shells]

[12] LFI: [https://www.invicti.com/learn/local-file-inclusion-lfi/]

[13] Preseed: [https://wiki.debian.org/DebianInstaller/Preseed]

[14] Creating a preconfiguration file: [https://d-i.debian.org/doc/installation-guide/en.amd64/apbs03.html]

[15] Contents of the preconfiguration file: [https://d-i.debian.org/doc/installation-guide/en.amd64/apbs04.html]

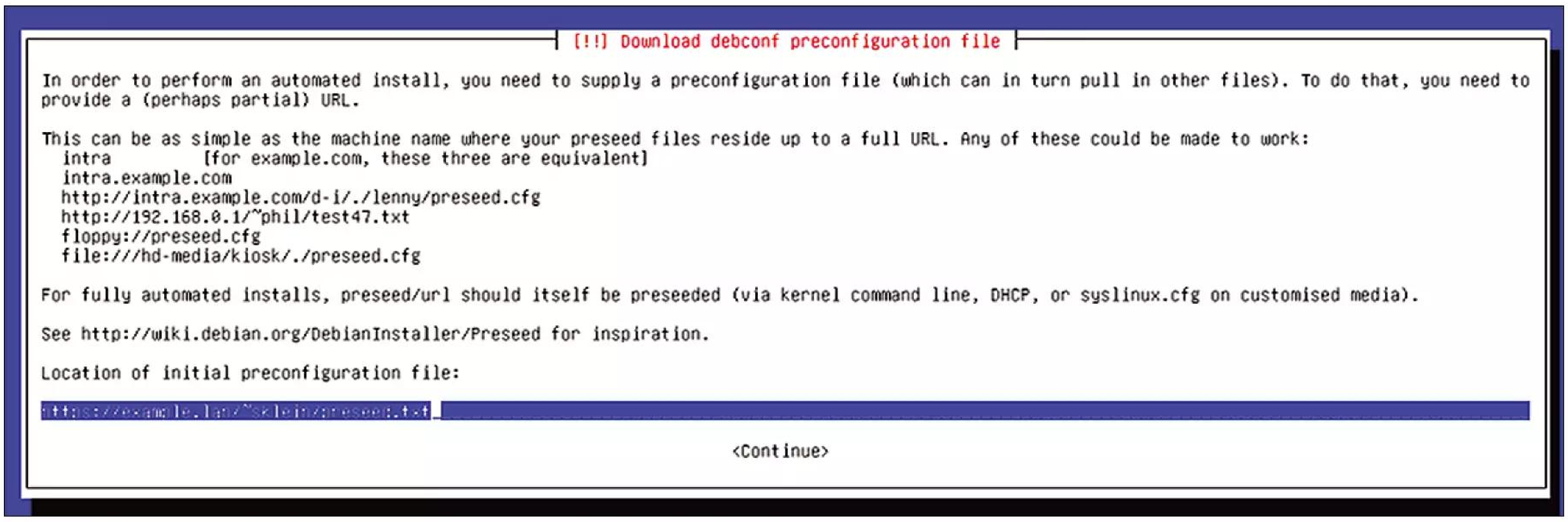
[16] Example preseed: [https://d-i.debian.org/doc/installation-guide/example-preseed.txt]

[17] Cubic: [https://launchpad.net/cubic]

[18] Simple ISO creation script: [https://gist.github.com/s-klein/1eba3b13988457bc883b52f55cb263ad]

The Author

Sam Klein is a cybersecurity engineer with more than six years of experience safeguarding enterprise infrastructure and shaping resilient systems at scale. His career spans embedded Linux platforms, open source development, and academic collaborations on privacy and application security. Klein is currently on sabbatical for full-time parenting while continuing to contribute to the cybersecurity community.



```
[!!] Download debconf preconfiguration file

In order to perform an automated install, you need to supply a preconfiguration file (which can in turn pull in other files). To do that, you need to
provide a (perhaps partial) URL.

This can be as simple as the machine name where your preseed files reside up to a full URL. Any of these could be made to work:
  intra         [for example.com, these three are equivalent]
  intra.example.com
  http://intra.example.com/d-i/./lenny/preseed.cfg
  http://192.168.0.1/~phil/test47.txt
  floppy://preseed.cfg
  file:///hd-media/kiosk/./preseed.cfg

For fully automated installs, preseed/url should itself be preseeded (via kernel command line, DHCP, or syslinux.cfg on customised media).

See http://wiki.debian.org/DebianInstaller/Preseed for inspiration.

Location of initial preconfiguration file:

https://example.lan/~sklein/preseed.txt_

<Continue>
```

Figure 10: An example of using a remote preseed file from the installer.

IT Highlights at a Glance

ADMINUPDATE

Issue 439 • July 23, 2025

This Week's Feature

Serverless Applications
OpenFaaS

Use advanced techniques for au
optimization, and security to dep
applications with the OpenFaaS
Linux environment.

News and Resources

- SUSE Virtualization 1.5 Prime Generally Available
- Rackspace Launches OpenStack Business
- AWS Previews Kiro Agentic IDE

In Case You Missed It

Data Protection Techniques

Data anonymization and pseudonymization are two key techniques that ensure privacy while enabling the use of data for analysis and decision making; the two methods offer different approaches that vary as a function of use case.

LINUX UPDATE

No. 419 • July 10, 2025

Linux 6.12 LTS at a Glance

For 20 years, the kernel developers maintained real-time support outside the mainline kernel. In Linux 6.12, real-time support has become an official part of the operating system kernel.

In the News

- Fedora Continues 32-Bit Support
- Bugs Found in sudo
- LibreOffice Tested as Possible MS Office Alternative

In Case You Missed It

- Compare PDFs with DiffPDF
  Most PDF viewers lack a function for comparing PDF files, but DiffPDF shows you the differences at a glance.
- Stats with Bash
  With just one line of Bash you can use tools like AWK and gnuplot to quickly analyze and plot your data.
- TRENDING: A Comprehensive Content Blocker
  This browser extension may be the ultimate tool for online privacy and security.

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox. Subscribe today for our excellent newsletters:

ADMIN Update • Linux Update

and keep your finger on the pulse of the IT industry.

ADMIN Update:
bit.ly/ADMIN-Update

Linux Update:
bit.ly/Linux-Update

Automating migration to openEuler

The Fast Migration Path

Easily migrate your old CentOS 7 systems to the modern openEuler with the x2openEuler migration toolset.

By Rubén Llorente

Photo by Chris Briggs on Unsplash

For years, CentOS was the distribution of choice for system administrators that needed to run a Red Hat Enterprise Linux (RHEL) compatible system, but could not afford a full support contract with Red Hat. CentOS, a fully compatible clone built from RHEL sources, functioned as a drop-in replacement for RHEL. CentOS promised an enterprise-grade Linux distribution that was available for anyone regardless of their budget.

Red Hat eventually became a sponsor for CentOS in 2014 **[1]**, in a movement the community found controversial at the time. The sponsorship agreement was, according to LWN.net **[2]**, a de facto acquisition of CentOS by Red Hat, since Red Hat took ownership of the CentOS trademarks and the CentOS governing board consisted of a majority of Red Hat employees.

In 2020, Red Hat unilaterally discontinued CentOS development, setting end-of-life (EOL) dates for the active versions much earlier than the community expected. According to Red Hat, CentOS users had no need to worry, because they could use the new CentOS Stream instead. Sadly, CentOS Stream is CentOS in name only. While the original CentOS was a RHEL drop-in replacement, CentOS Stream is an upstream development version of Red Hat, which made serious users wary of migrating.

The last CentOS supported release, CentOS 7, reached EOL on June 30, 2024. If you are still using CentOS today, you are running an unsupported operating system. The good news is that many alternatives have arisen in response to CentOS' demise, so system administrators have plenty of migration options.

One such option, openEuler from the OpenAtom foundation, offers an Enterprise Linux-like system for servers and cloud environments. It boasts a number of unique utilities, such as iSulad, a lightweight container runtime daemon; A-Tune, an AI-powered performance tuning tool for servers; and the x2openEuler migration tool. In this article, I explain how to use x2openEuler to migrate a small fleet of CentOS 7 servers to openEuler.

Your Migration Plan

Traditionally, upgrading systems from the RHEL family involved recreating your services on the new operating system, redeploying them from scratch, and discarding the old installations. For complex services with tight dependencies that cannot easily be carried over from your old CentOS to the new openEuler, this route might still be required. However, the vast majority of applications can be upgraded through different means.

The ideal, modern way of carrying out a migration is via in-place upgrades. When performing an in-place upgrade, you access each server that you want to migrate, replace the operating system that is running under your applications, and keep your production services pretty much untouched. The advantage of this approach is that it is fast, requires less effort, and lets you reuse your old service configuration.

As a rule of thumb, if the platform your production services run on is available in openEuler's repositories, an in-place migration is the way to go. For example, if your application runs on a modern OpenJDK, Perl, or Python environment, chances are an in-place migration will work free of fuss. To see if the platform your service runs on is available, check Repology **[3]** to verify that openEuler has the packages you need.

I strongly advise backing up your servers before you attempt to upgrade them. While the migration suite has a rollback functionality, you should not count on your migration system to safeguard your data. Ensure you have a sound backup before you proceed.

In-place migrations require the servers to reboot at certain points, therefore you should be prepared for service interruption during the process.

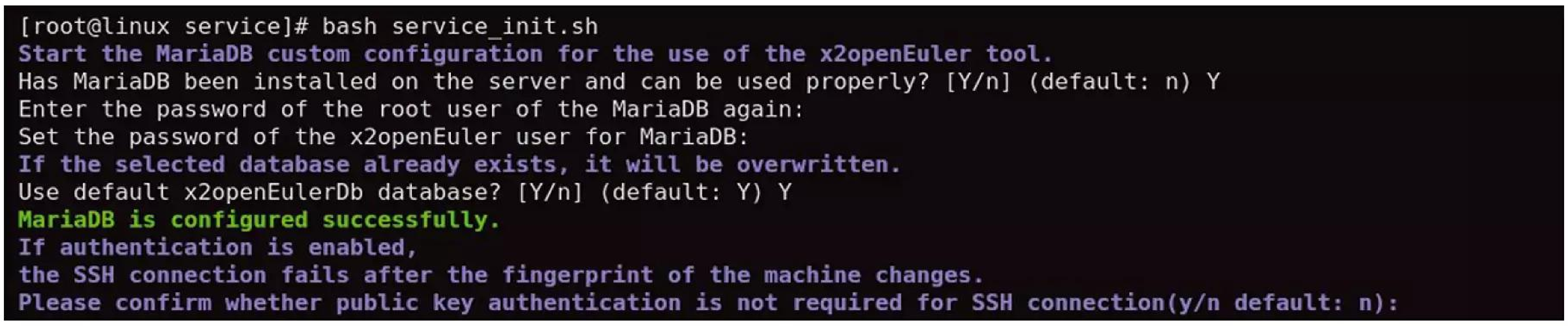
Listing 1: Setting the MariaDB Root Password

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('your_password_here');
FLUSH PRIVILEGES;
QUIT;
```

Your Migration Suite

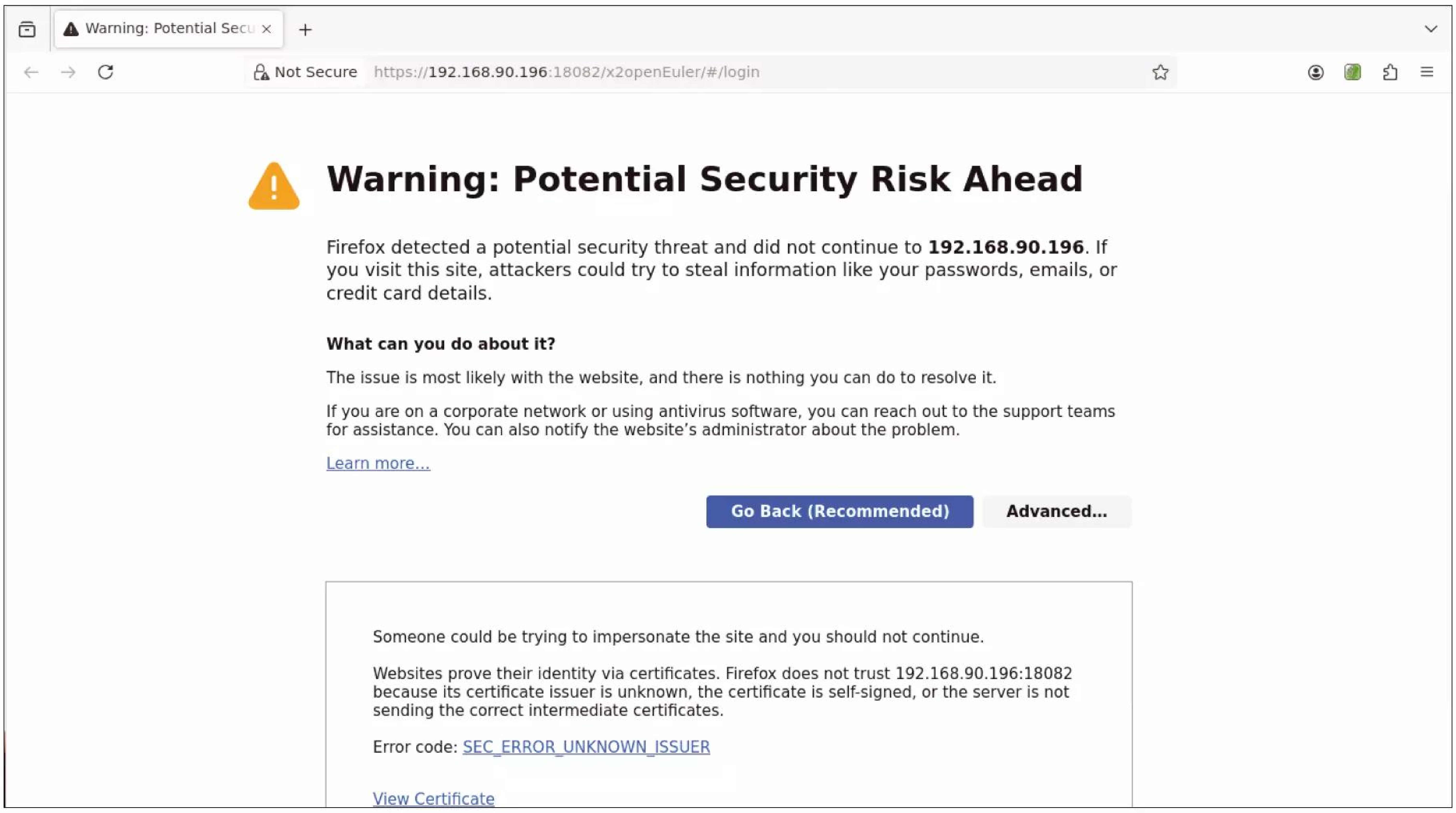
The x2openEuler upgrade management tool needs to be installed on a machine in your network (preferably a server), be configured by an administrator, and then be instructed to perform a number of operations on your server fleet. x2openEuler operates by opening SSH sessions from the control machine to connect to your target servers. This design makes x2openEuler a fine option for migrating small- and medium-sized server fleets, because the administrator can install x2openEuler on a single control node, provide it with the all the target servers' addresses and SSH credentials, and have it migrate them all automatically. Servers can also be migrated in groups if you don't want to switch all your infrastructure in one go – you can also choose to migrate in small increments instead.

To install x2openEuler, grab it from the openEuler download page **[4]**. x2openEuler is available as an RPM package for the x86_64 and AArch64 architectures. If your goal is to migrate an old server fleet



```
[root@linux service]# bash service_init.sh
Start the MariaDB custom configuration for the use of the x2openEuler tool.
Has MariaDB been installed on the server and can be used properly? [Y/n] (default: n) Y
Enter the password of the root user of the MariaDB again:
Set the password of the x2openEuler user for MariaDB:
If the selected database already exists, it will be overwritten.
Use default x2openEulerDb database? [Y/n] (default: Y) Y
MariaDB is configured successfully.
If authentication is enabled,
the SSH connection fails after the fingerprint of the machine changes.
Please confirm whether public key authentication is not required for SSH connection(y/n default: n):
```

Figure 1: The x2openEuler initialization process.



Warning: Potential Secu ×
Not Secure
https://192.168.90.196:18082/x2openEuler/#/login
Go Back (Recommended)
Advanced...

Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to **192.168.90.196**. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

What can you do about it?

The issue is most likely with the website, and there is nothing you can do to resolve it.

If you are on a corporate network or using antivirus software, you can reach out to the support teams for assistance. You can also notify the website's administrator about the problem.

Learn more...

Someone could be trying to impersonate the site and you should not continue.

Websites prove their identity via certificates. Firefox does not trust 192.168.90.196:18082 because its certificate issuer is unknown, the certificate is self-signed, or the server is not sending the correct intermediate certificates.

Error code: SEC_ERROR_UNKNOWN_ISSUER

View Certificate

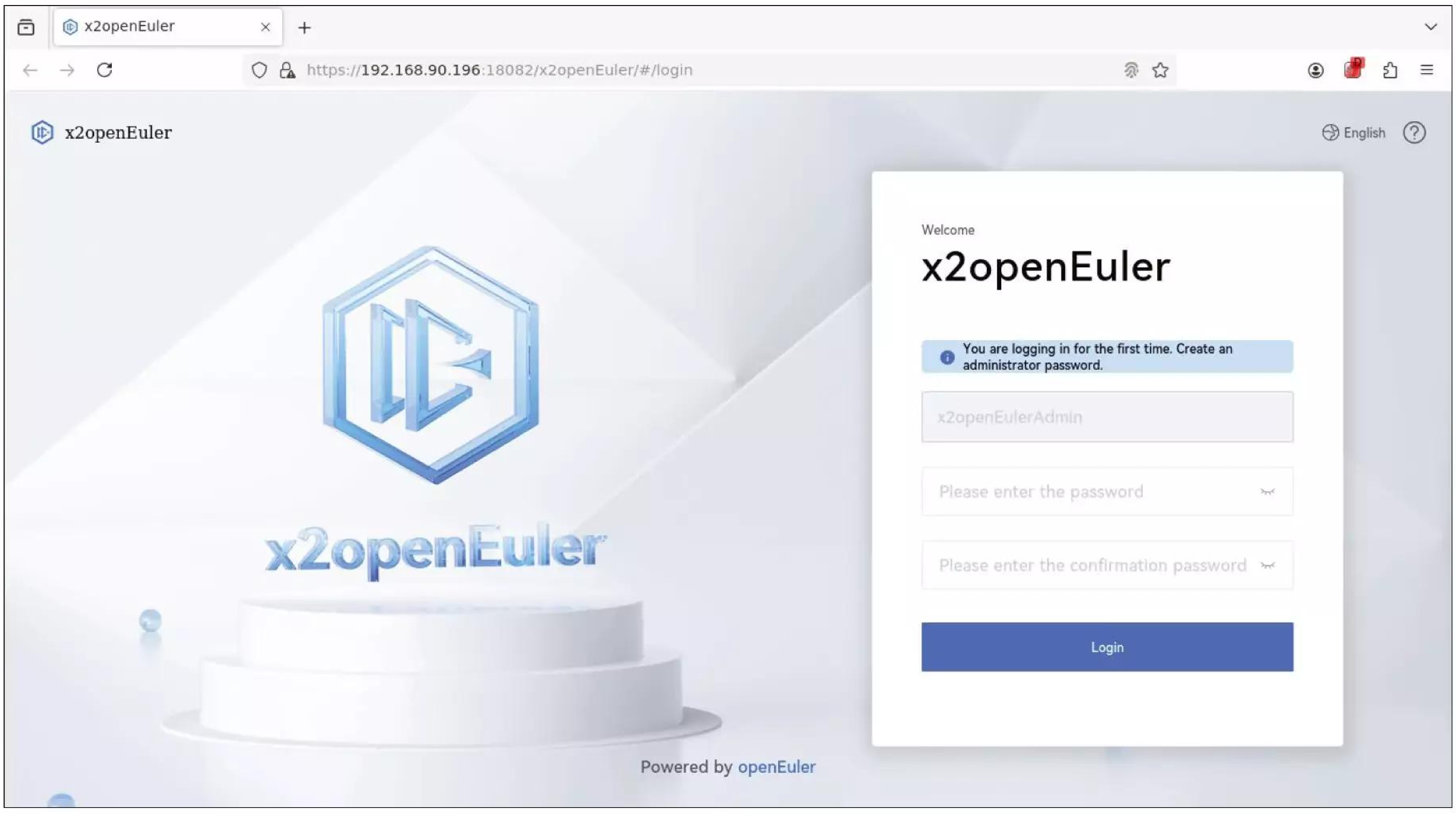
Figure 2: The migration suite uses a self-signed certificate. As a result, the web browser will throw a warning the first time you access it.

composed of CentOS 7 systems, it's safe to assume you will be installing x2openEuler on a CentOS 7 control node. I recommend running x2openEuler from an instance devoted solely to this task that you can discard after the migration is complete. In order to install x2openEuler, just upload the RPM package to the machine and install it with `yum`:

```
# yum install ./x2openEuler-core*.rpm
```

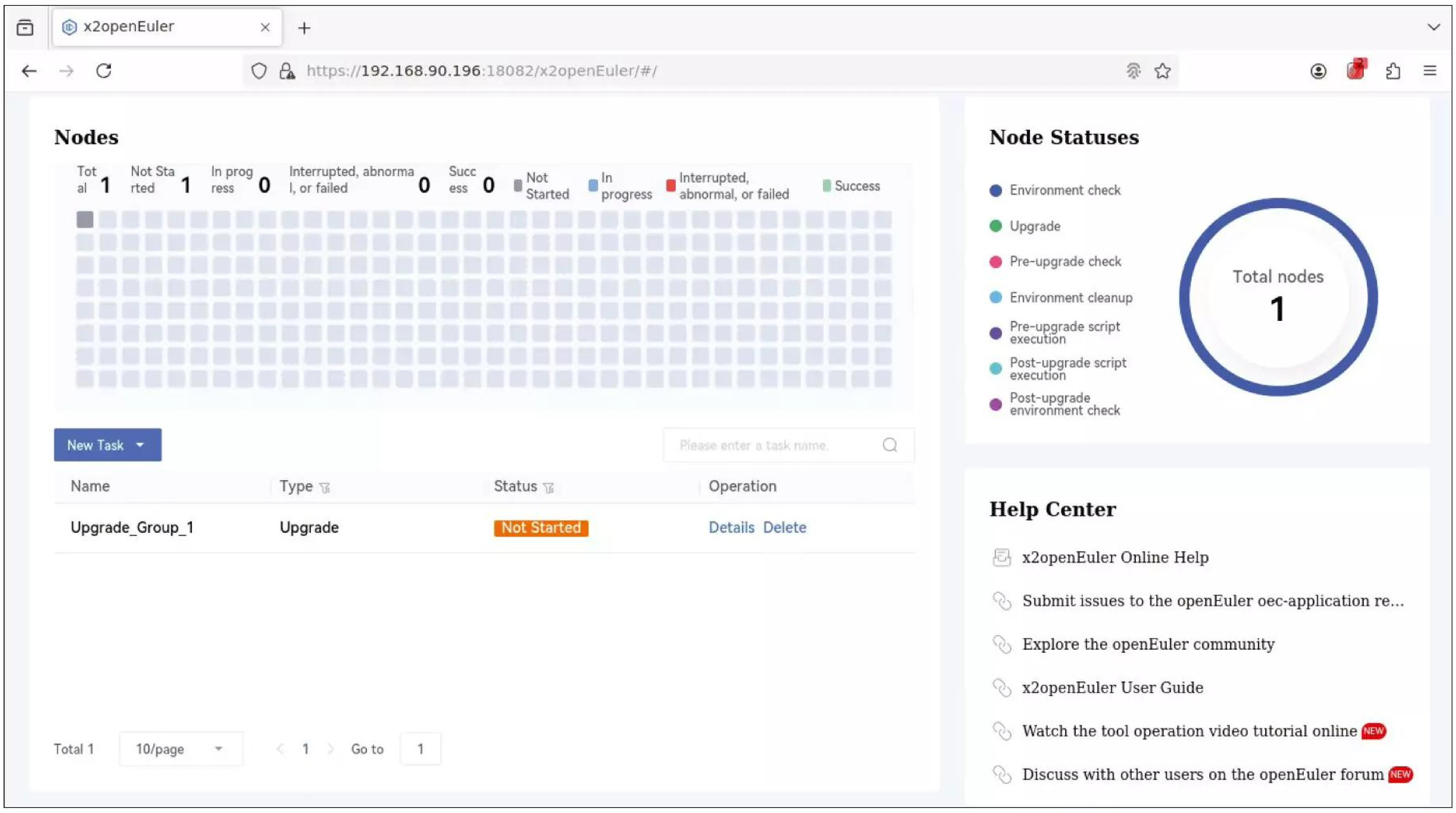
The installation process pulls a surprising number of dependencies with it, including the MariaDB database (as well other databases). You should set a root password for your database to make the initialization of x2openEuler smoother. Just start the database service with

```
# systemctl start mariadb
```


x2openEuler
https://192.168.90.196:18082/x2openEuler/#/login
x2openEuler
English
Welcome
x2openEuler
x2openEulerAdmin
Please enter the password
Please enter the confirmation password
Login
x2openEuler
Powered by openEuler

You are logging in for the first time. Create an administrator password.

Figure 3: You must create an administrator user in order to log in for the first time. The web interface supports both the Chinese and English languages, which you can change from a menu at the top right of the page.


x2openEuler
https://192.168.90.196:18082/x2openEuler/#/
Nodes
Total 1
Not Started 1
In progress 0
Interrupted, abnormal, or failed 0
Success 0
Not Started
In progress
Interrupted, abnormal, or failed
Success
New Task
Please enter a task name.
Total 1
10/page
1
Go to
1
Node Statuses
Environment check
Upgrade
Pre-upgrade check
Environment cleanup
Pre-upgrade script execution
Post-upgrade script execution
Post-upgrade environment check
Total nodes
1
Help Center
x2openEuler Online Help
Submit issues to the openEuler oec-application re...
Explore the openEuler community
x2openEuler User Guide
Watch the tool operation video tutorial online NEW
Discuss with other users on the openEuler forum NEW

Name	Type	Status	Operation
Upgrade_Group_1	Upgrade	Not Started	Details Delete

Figure 4: The task manager is the core of x2openEuler. Use the *New Task* menu to create a new *Upgrade* task.

Using the `mysql` command, launch a database prompt and set your desired password for the database root user. The root user here is the root user within the database context only – these actions don't modify your system root access credentials! See Listing 1 for the command sequence to set your database root password. Your password must feature a combination of lowercase, uppercase, and special characters; otherwise, it will be rejected.

Next, you need to initialize x2openEuler itself. Within your terminal session, move to `/usr/local/x2openEuler/portal/service/` and get started using the script `service_init.sh`. You will be asked a number of questions regarding your setup (Figure 1); feel free to use common-sense answers.

The initialization script will perform a number of tasks in order to enable and launch the service, including the generation of a self-signed TLS certificate for x2openEuler's web interface. At the end of the process, the server will be running, and you will be able to access the web interface by pointing a web browser to *https://$yourserver:18082/x2openEuler/#/login* (where *$yourserver* is the machine's IP address).

Because x2openEuler uses a self-signed certificate, you should expect your web browser to throw a warning (Figure 2) before it lets you access the web interface (Figure 3).

Also worth mentioning, the x2openEuler service will be vulnerable until an administrator logs in for the first time. Therefore, I recommwend keeping the machine that runs the service behind a firewall with no connection to outside of your organization.

The Migration Targets

x2openEuler is a surprisingly complete tool. If you are operating within a serious environment, your first stop will probably be *Web Server Certificates* (accessible from the gear on the top-right corner of the administration dashboard). From here, you can replace the default self-signed certificate with one that is widely recognized by your organization's web browsers. Additional users can be added via the *User Management* section, in case you need to grant access to your coworkers.

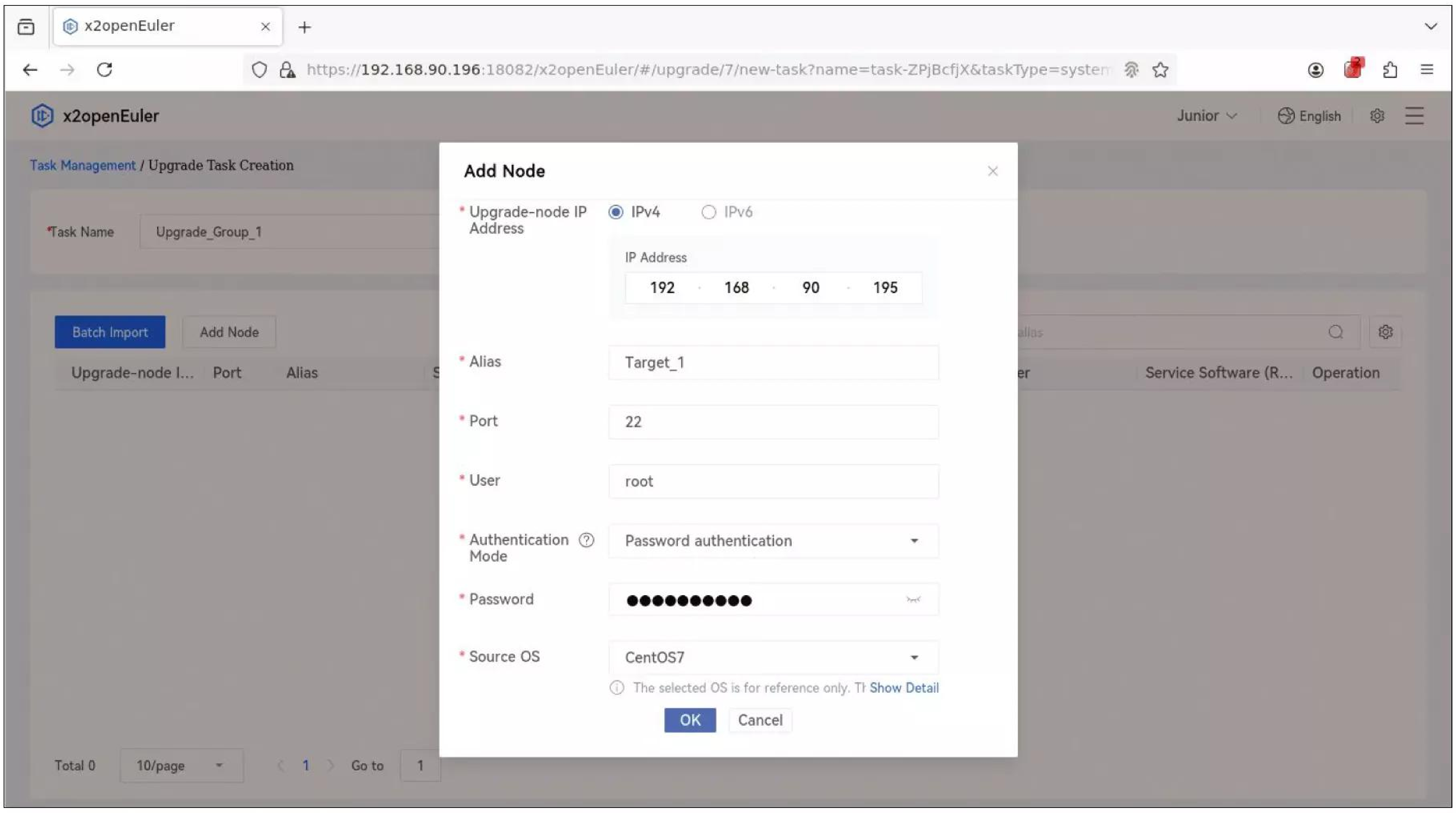
Task management is x2openEuler's main feature (Figure 4). When you open the task management interface, you'll be presented with many things to do. If the only thing you need for the moment is to upgrade your server fleet, you can cut to the chase and hit *New Task* to create a new *Upgrade* task.

Within the task creation section, you may add as many target servers as you like. You may either add them in a batch by uploading a Microsoft Excel file (a template file is provided) or you can add servers one by one. x2openEuler will manage the target servers over an SSH connection, and both password- and keyfile-based authentication are supported (Figure 5).

From the same menu where the target servers are defined, you can also configure a list of packages you want to be installed on the migrated system later, run custom scripts at different phases of the process, and perform a number of advanced actions.

After you have added all the servers you want to migrate, hitting *Ok* at the bottom of the task page will start the migration task. As a first step, it will begin performing preliminary environment tests. You will be taken to a status page that will show a global report of the ongoing testing. From here, you can check individual servers for an in-depth detailed view of their status (Figure 6).

The whole migration cycle for each server consists of four phases: an *Environment check*, a *Pre-upgrade*



https://192.168.90.196:18082/x2openEuler/#/upgrade/7/new-task?name=task-ZPjBcfjX&taskType=system

Task Management / Upgrade Task Creation

*Task Name Upgrade_Group_1

Add Node

* Upgrade-node IP Address: ● IPv4 ○ IPv6

IP Address: 192 . 168 . 90 . 195

* Alias: Target_1

* Port: 22

* User: root

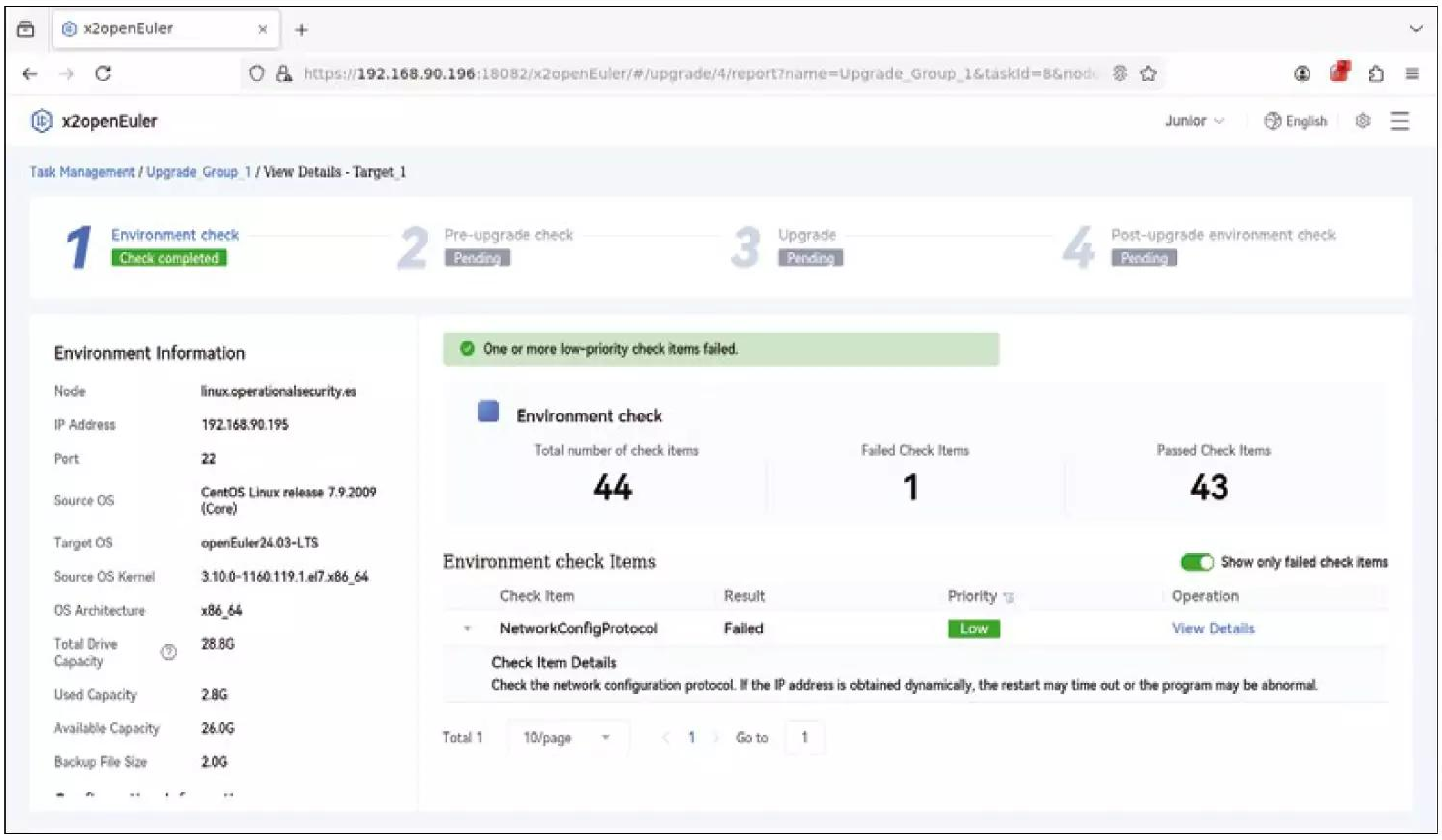
* Authentication Mode: Password authentication

* Password: ●●●●●●●●●●

* Source OS: CentOS7

The selected OS is for reference only. Tl Show Detail

Figure 5: Servers can be added to your upgrade task one by one. If this method is too slow, you can also import a list via a Microsoft Excel file.



x2openEuler

https://192.168.90.196:18082/x2openEuler/#/upgrade/4/report?name=Upgrade_Group_1&taskId=8&nod

x2openEuler

Junior English

Task Management / Upgrade_Group_1 / View Details - Target_1

1 Environment check Check completed

2 Pre-upgrade check Pending

3 Upgrade Pending

4 Post-upgrade environment check Pending

Environment Information

Node	linux.operationalsecurity.es
IP Address	192.168.90.195
Port	22
Source OS	CentOS Linux release 7.9.2009 (Core)
Target OS	openEuler24.03-LTS
Source OS Kernel	3.10.0-1160.119.1.el7.x86_64
OS Architecture	x86_64
Total Drive Capacity	28.8G
Used Capacity	2.8G
Available Capacity	26.0G
Backup File Size	2.0G

One or more low-priority check items failed.

Environment check

Total number of check items	Failed Check Items	Passed Check Items
44	1	43

Environment check Items

Show only failed check items

Check Item	Result	Priority	Operation
NetworkConfigProtocol	Failed	Low	View Details

Check Item Details

Check the network configuration protocol. If the IP address is obtained dynamically, the restart may time out or the program may be abnormal.

Total 1 10/page 1 Go to 1

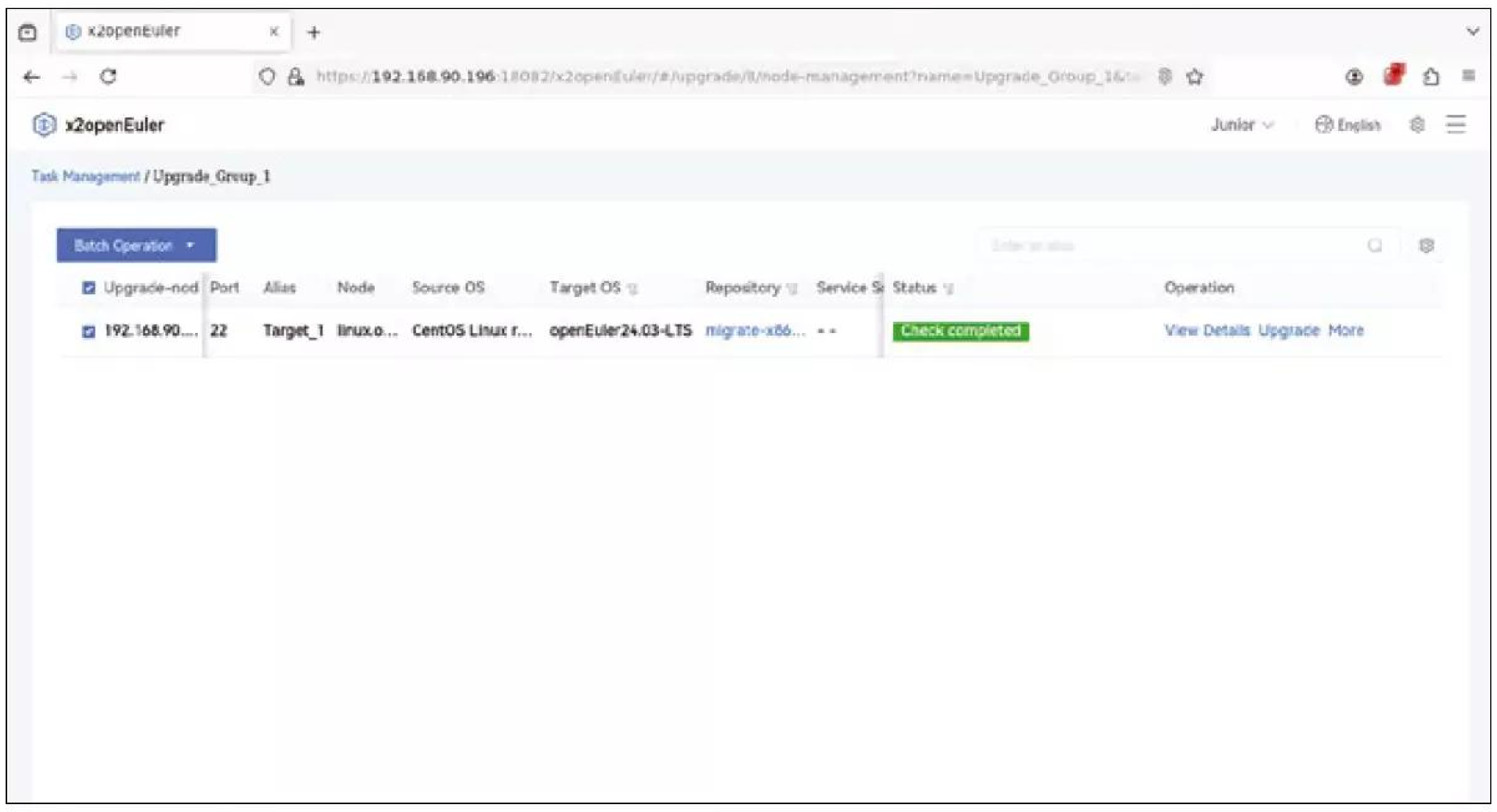
Figure 6: The detailed status for one of the servers: This migration task shows a minor failure because x2openEuler thinks this server might be using a dynamic IP address.

check, the actual *Upgrade*, and the *Post-upgrade environment check*. You can trigger these for each server from the *Task Management* tab or by calling a batch action that includes all the targeted servers (Figure 7).

Once the *Upgrade* phase is complete, you are given a final chance to rollback each server to the previous state. x2openEuler has the capability to rollback to the pre-migration stage, and you can go back using the *Rollback* button on the bottom right of the screen. If you are satisfied with the state of the server after the migration, you can move on to running the post-upgrade checks.

Conclusion

Migrating away from an obsolete CentOS installation doesn't have to be hard – it can actually be a quick process. You should expect a small server to be fully migrated in little more than an hour. If you're looking for a Enterprise Linux alternative that isn't focused on 1:1 compatibility with IBM's offering and that is actively developing their own cloud and server solutions, openEuler is worth considering, if only because their migration and upgrade tools are so powerful and convenient. This is not to say that x2openEuler cannot be improved. Something that surprised me a bit, for example, was that users can't share migration tasks. x2openEuler users can only create tasks for themselves and can only execute their own tasks. While not a deal breaker, I think openEuler is missing the opportunity to make x2openEuler a true collaborative suite with granular permissions – useful, for example, if you want a junior system administrator to create the migration tasks and run the preliminary checks, but want a senior sys admin to trigger the actual upgrade. ■



x2openEuler

https://192.168.90.196:18082/x2openEuler/#/upgrade/8/node-management?name=Upgrade_Group_1&

x2openEuler

Junior English

Task Management / Upgrade_Group_1

Batch Operation

☑ Upgrade-nod	Port	Alias	Node	Source OS	Target OS	Repository	Service S	Status	Operation
☑ 192.168.90....	22	Target_1	linux.o...	CentOS Linux r...	openEuler24.03-LTS	migrate-x86...	--	Check completed	View Details Upgrade More

Figure 7: A task with a single target server: You can trigger the *Upgrade* phase by clicking on *Upgrade*. If a task contains more than one server, you can select *Batch Operation* to upgrade them all at once.

Info

[1] "CentOS Project joins forces with Red Hat": [https://lists.centos.org/pipermail/centos-announce/2014-January/020100.html]

[2] "CentOS and Red Hat team up": [https://lwn.net/Articles/579551/]

[3] Repology: [https://repology.org]

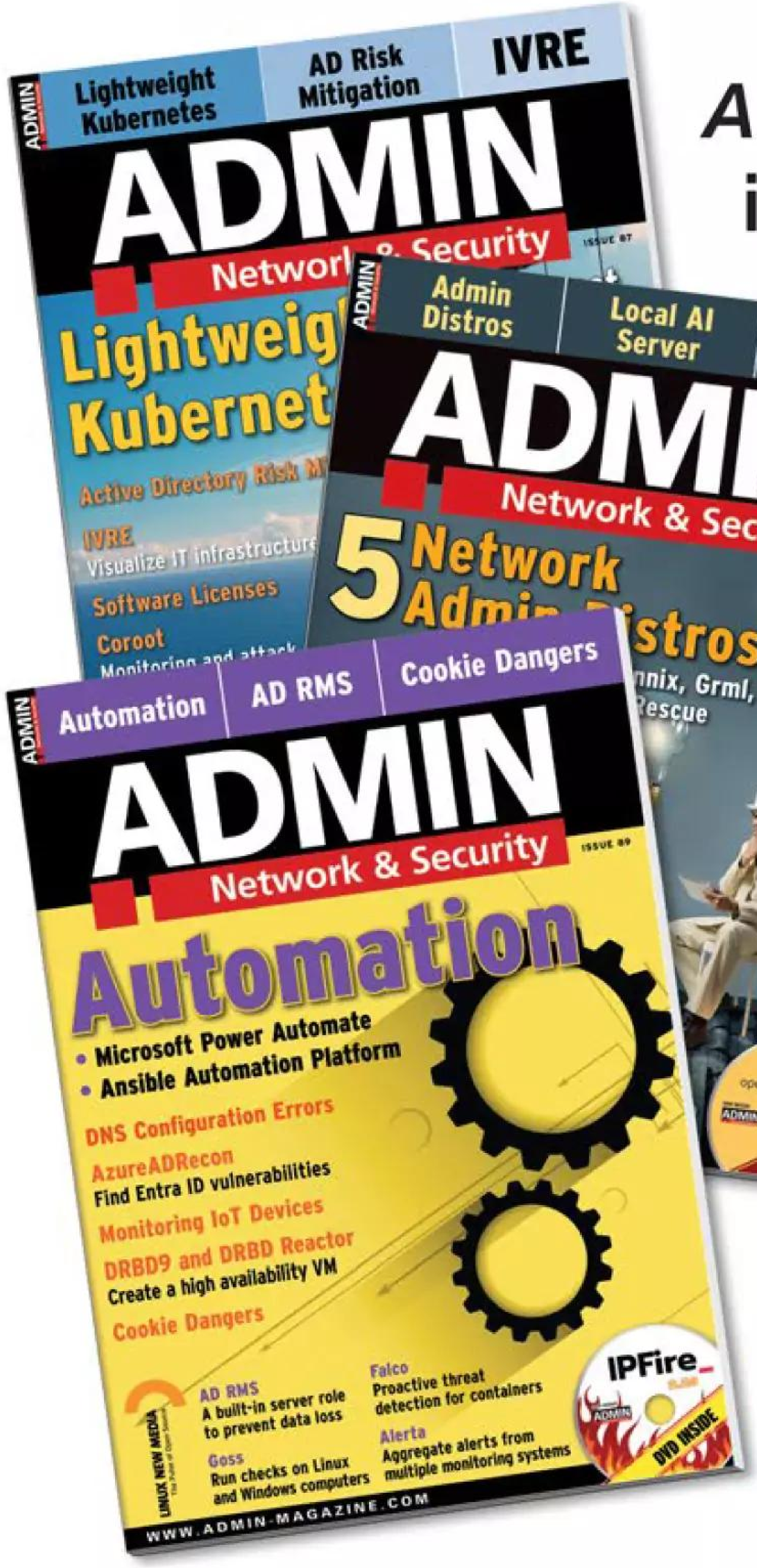
[4] x2openEuler download: [https://www.openeuler.org/en/migration/download/]

This article was made possible by support from openEuler by OpenAtom Foundation through Linux New Media's Topic Subsidy Program (https://www.linuxnew-media.com/Topic_Subsidy).

The Author

Rubén Llorente is a mechanical engineer whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both legally compliant and safe. He is also an OpenBSD enthusiast.

GET TO KNOW ADMIN


Lightweight Kubernetes
AD Risk Mitigation
IVRE
ADMIN Network & Security
Lightweight Kubernetes
Active Directory Risk M
IVRE Visualize IT infrastructure
Software Licenses
Coroot
Admin Distros
Local AI Server
A New Checkmk
ADMIN Network & Security
5 Network Admin Distros
openSUSE Leap 15.6
DVD INSIDE
Automation
AD RMS
Cookie Dangers
ADMIN Network & Security
Automation
Microsoft Power Automate
Ansible Automation Platform
DNS Configuration Errors
AzureADRecon Find Entra ID vulnerabilities
Monitoring IoT Devices
DRBD9 and DRBD Reactor Create a high availability VM
Cookie Dangers
AD RMS A built-in server role to prevent data loss
Goss Run checks on Linux and Windows computers
Falco Proactive threat detection for containers
Alerta Aggregate alerts from multiple monitoring systems
IPFire
DVD INSIDE
WWW.ADMIN-MAGAZINE.COM

ADMIN Network & Security **magazine is your source for technical solutions to real-world problems.**

ADMIN is packed with detailed discussions aimed at the professional reader on contemporary topics including security, cloud computing, DevOps, HPC, containers, networking, and more.

Subscribe to *ADMIN*

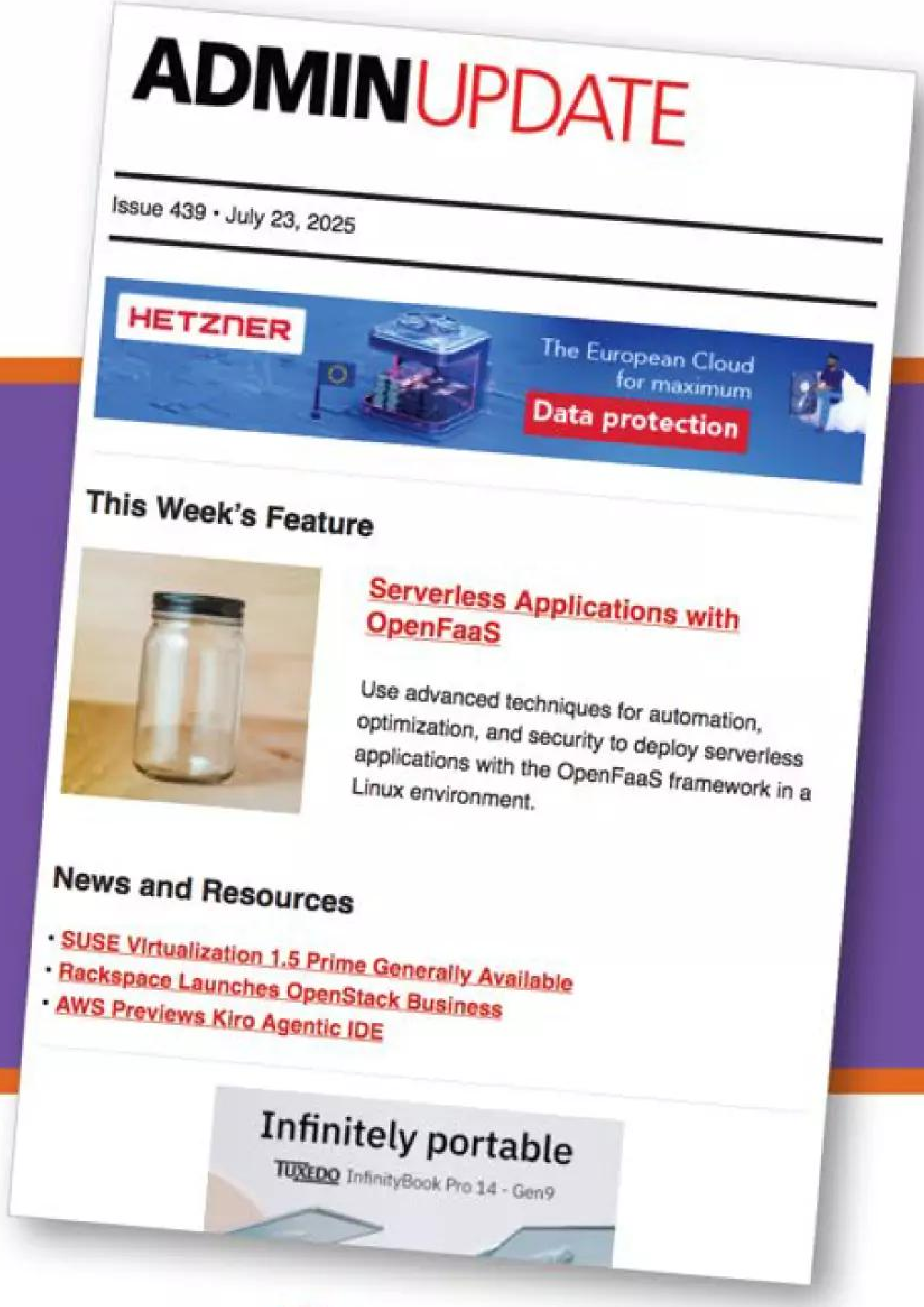
and get 6 issues delivered every year

Want to get ADMIN in your inbox?

Subscribe free to ADMIN Update

and get news and technical articles you won't see in the magazine.

bit.ly/ADMIN-Update


ADMINUPDATE

Issue 439 • July 23, 2025

HETZNER
The European Cloud for maximum
Data protection

This Week's Feature

Serverless Applications with OpenFaaS

Use advanced techniques for automation, optimization, and security to deploy serverless applications with the OpenFaaS framework in a Linux environment.

News and Resources

- SUSE Virtualization 1.5 Prime Generally Available
- Rackspace Launches OpenStack Business
- AWS Previews Kiro Agentic IDE

Infinitely portable
TUXEDO InfinityBook Pro 14 - Gen9

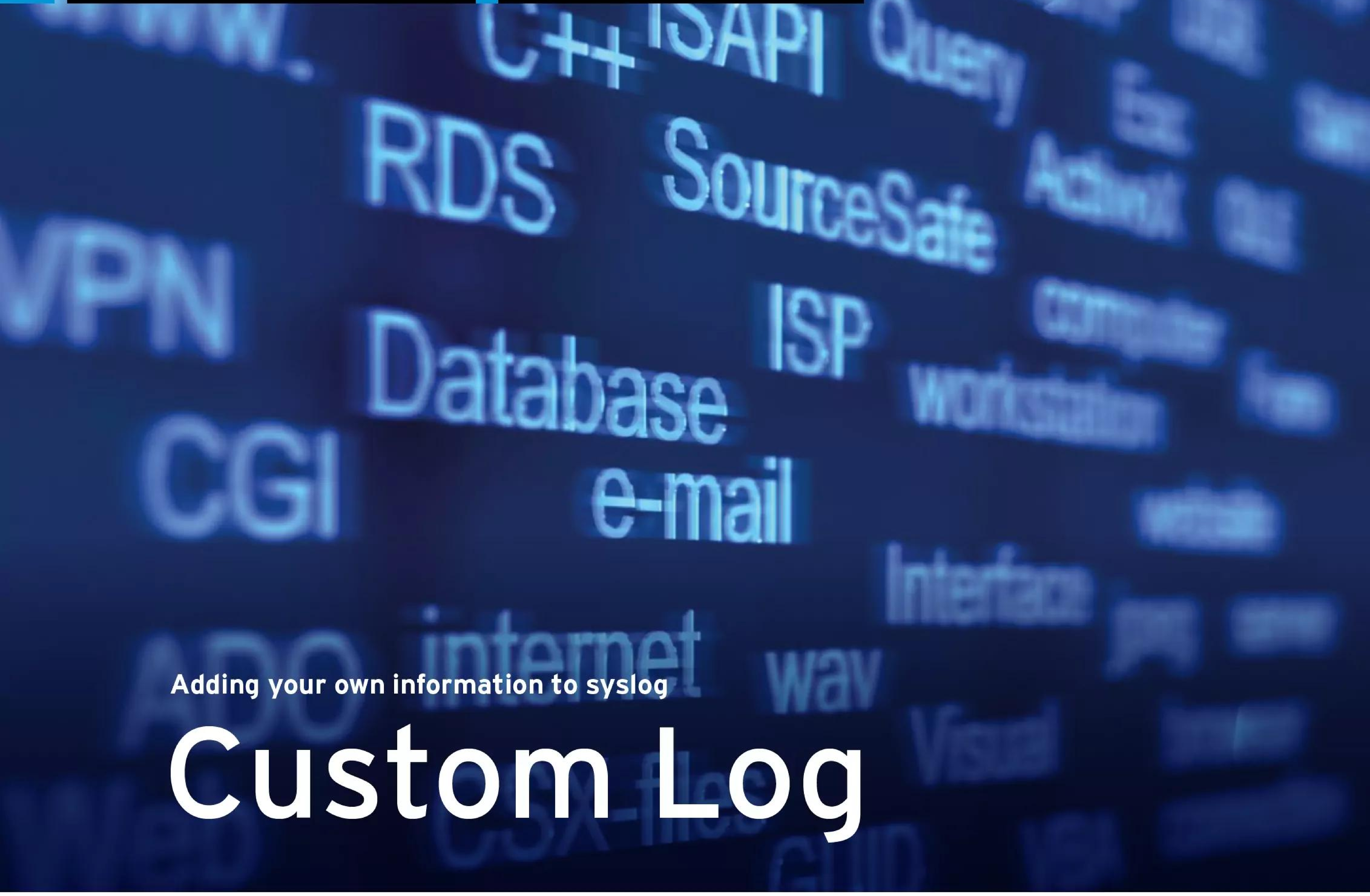
ADMIN
Network & Security

@admin-magazine.com

ADMIN magazine

@adminmag

@adminmagazine@hachyderm.io



Adding your own information to syslog

Custom Log

With logger, you can store user-specified messages in syslog to gather information for HPC admin tasks.

By Jeff Layton

HPC administrators periodically look through the system logs in `/var/log`, searching for errors or other interesting information. Although it is very nice that the logs are centralized and that admins have tools that use these logs to provide notifications, they are not easy to process and understand. Wouldn't it be nice to be able to put your own information into the system log (`/var/log/syslog`) so you can parse it for your messages with common tools such as `grep`? Of course, Linux has a tool for that: `logger`.

The system log is a central location for all system messages and would be a convenient location for storing your own messages. Any automatic log processing tool will catch these messages, but you can easily set up a simple `grep` to capture the messages you publish to `syslog` either manually or in scripts. This capability allows you to use all kinds of logging as part of your system administration, including adding `logger` entries into the system log. The `logger` entries can be done manually or incorporated into scripts.

Introduction to Logger

All of the examples in this article are from a rather dated Ubuntu 20.04 system, although `logger` works the same way on current distributions. The basic syntax to write a simple message to the system log is pretty easy:

```
$ logger "Just a test"
```

The entry in `syslog` will include a timestamp indicating when the message was written to the system log, the name of the system, the user who wrote the message or the tag used to write the message, and, after a colon, the message itself:

```
...
Jun 28 12:24:55 laytonjb-Nitro-AN515-55 ➘
  laytonjb: Just a test
...
```

This capability is really simple but very powerful. You can create the message as part of a script (e.g., a Bash or Python script), so you can construct a message that includes important details. The details could include the name of the script along with its purpose, perhaps the input to script (if any), output from the script (e.g., the status or location within the script, specifically the line number), and perhaps the result of the script (whether or not it succeeded).

If you use `logger` manually, you can write a message to the system log for anything you want. Perhaps you can write a message just before you start writing code or to check the status of the system. Make the message something useful so you can identify its purpose in the log; more importantly, an entry in `syslog` with a timestamp tells you when you started. When you finish the project, you can use `logger` to add another entry saying you completed the project, including any interesting notes that would be useful when coming back to the project.

To find the elapsed time, use `grep` to look for the messages. For example, if you're working on a project named

Lead Image © Andrey Prokhorov, Fotolia.com

Orion-Meatloaf, you just search for that project name:

```
$ sudo grep -i "project orion-meatloaf" ↩
  /var/log/syslog
```

This command shows all entries in the current system log for that project. (I like to include `project` in the name to differentiate the message from administrative tasks, such as reading other logs (e.g., `/var/log/auth.log` or `/var/log/boot.log`) or performing other specific admin tasks – including debugging problems.

Permissions

The user account on my laptop allows me to write to but not read from `syslog`. You might encounter a user account that won't allow them to write to `syslog`. You will have to give that user permission to do so if they need it. However, who you allow to read `syslog` is up to you. You could give a user `sudo` permission to `syslog` if you want, or you could sanitize `syslog` by removing all but the user's messages before handing over the data to the user (this task also can be scripted).

Tagging

If `logger` is used from the command line with the default options, you will get the user's name with the message. Sometimes, however, the user might want to add a specific tag to the message:

```
$ logger -t myscript "A second test"
```

The entry in the system log would look like:

```
$ sudo tail -n 10 /var/log/syslog
...
Jun 28 14:28:55 laytonjb-Nitro-AN515-55 ↩
  myscript: A second test
...
```

Notice that rather than using the user's name, it uses the tag `myscript`. This way of adding specific information to the messages in `syslog` is particularly helpful with scripts that are in cron jobs or other automated scripts.

Linux pipes can also be used to create complex commands, such as,

```
$ echo "Output redirect" | logger
```

to redirect the output from a command to `logger`.

Logging In

Whenever a user logs in to a system, it is logged in `/var/log/auth.log`, but you could easily use `logger` to add a script to a user's account or perhaps create a script that wraps the login binary (`/bin/login`) that adds messages to `syslog`. You might be duplicating what `/var/log/auth.log` does, but you can add whatever extra information you want in the wrapper script and add messages to the system log.

Environment Modules

In the HPC world, environment modules allow users to control their environment, such as changing compilers, libraries, and other tools, without having to change the user `$PATH`, `$LD_LIBRARY_PATH`, and so on. Much of the environment module implementations use scripts when the user changes modules or purges the loaded modules, which means it can all be captured and sent to the system log. The two most common environment module tools, `modules` and `Lmod`, capture much of this information, but with logger you can add to the system log whatever information you like, need, or want. Just be cognizant of the size of the system log if you have a large number of users that use environment modules.

Cron Jobs

Cron jobs are the perfect place to use `logger`. The cron service typically runs scripts that perform one or more tasks at specified time intervals. By default, standard output and standard error are both sent to the email of the user who owns the cron job. In the cron job script, you can add information to the standard output, such as the progress of the script, any issues (errors) that might have arisen, and any final output that ends up in an email. With `logger`, this data can be sent to the system logs, which, in my opinion, is easier to manipulate than email.

For example, I have a script that walks user home directories, gathering metadata information about their files. It doesn't access the files, but it accesses the metadata with the `stat` command. I use this information for a variety of purposes:

- To watch the increase in total space used by a particular user (uses previously gathered data)
- To find the oldest file for a particular user (I'm looking for stale data that could be moved to archival storage)
- To find the largest file for a particular user
- To count the number of small files for that user (I define "small" in the script as less than 4kB)

The script gathers the data and writes it to a CSV file for postprocessing, so I can find the information I want. I could send the raw data to `syslog`, but that's a very large amount of data sitting in that file. Instead, I can have the script just find the answers I want and then send a message with `logger` to `syslog`, such the following examples:

User X has a file <fully qualified path to file> that is N days old. The oldest file for User X is NN days.

I could add a tag to the message (e.g., `file_walking`) so that when I parse the system log, I could search for messages with this tag and then pipe the results through another `grep` command to output only data from the day of the search. Granted, I could do this in the original script and output to a particular file or send by email, but again, having it in a central location and being able to correlate `logger` messages with other messages in the system logs is very useful (in my opinion).

System Information

Also in the cron job category, but worth mentioning separately, is a script in a cron job that gathers system information such as CPU, memory, network, and storage usage or any other system measurement you want. After gathering that data, you could use `logger` to send it to the system log, where you now have information as to the status of the system. Normally, this kind of data is not captured and stored anywhere, but you can easily add that capability.

Of course, I don't think I would capture that information every second because the poor system log would explode. However, you can gather system information at a predefined interval to give you an idea of the system status. For example, maybe you could gather the system load every 15 minutes with `uptime` to get a very simplified view of the total system load. You could also grab information from the job scheduler (workload manager), such as how many jobs are running, how many jobs are waiting, how many nodes are free, and so on. Again, this is just a snapshot of the system status, but gathering data every 15 minutes means you are gathering 96 data points a day. Over a 30-day period, this is 2,880 data points, which can give you some good insight as to how the system is being used.

Logs Get Huge Quickly

A consideration you need to address before committing to `logger` is how much data will be added to the system log. The current `/var/log/syslog` is rotated out to `/var/log/syslog.1`, so a new syslog can begin. All of the previous syslogs have their extensions incremented by one, and the `syslog` with the largest count (e.g., `syslog.7`) is erased from the system, so `/var/log` won't get filled up. Also, any `syslog` that has a numerical extension is compressed to save space. Using `logger` means logs will be bigger, perhaps not by much, so you need to keep that in mind.

If you need the older system logs, which you might, you need to make arrangements for them to be copied to another filesystem. Be sure this secondary filesystem has enough space for as many logs as you need to keep. System logs are text, so you can get a very high compression ratio on them.

Another option is just to parse `syslog` for whatever information you need (e.g., `logger` messages), resulting in a much smaller amount of data that can be stored either on the system or in secondary storage. Because the files are small, you can keep them on the primary system much longer before shipping them to secondary storage.

Interaction with Remote Logs

HPC systems will often use remote logging on the compute nodes, which means the logs from all of the compute nodes will be sent to a single server. I have not tested how this interacts with `logger` because my home lab clusters are so small, but just in case, `logger` has the ability to send the messages directly to a remote server.

You can take advantage of this feature and send `logger` messages to a central server but leave all other system messages on the local server. In a way, it would be like filtering `syslog` and sending the `logger` messages to a central `syslog`.

Summary

I'm a big believer in logging lots of information, but if you don't actively use the data, it's pointless to have it in logs. The Linux tool `logger` allows you to create the messages you want and store them in the system log. You can define what is in the message, add tags, and specify when to send the message to `syslog`.

System administrators can do a great deal with `logger` to gather information, such as how time is spent working on projects. Scripts can gather just a little bit of information about how the user is functioning. I'm not a fan of logging too much user detail, and I will not gather any data that can be considered private, sensitive, or compromising to someone's work.

The Linux `logger` tool is also very useful for sys admin tasks that have scripts. Messages to `syslog` for these scripts that describe progress, errors, and results, is very simple to add. This additional data can greatly help debugging issues and can track results, as well. ■■■

The Author

Jeff Layton has been in the HPC business for over 30 years (starting when he was 4 years old). When he's not grappling with a stubborn systemd script, he's looking for deals for his home cluster. His twitter handle is @JeffdotLayton.

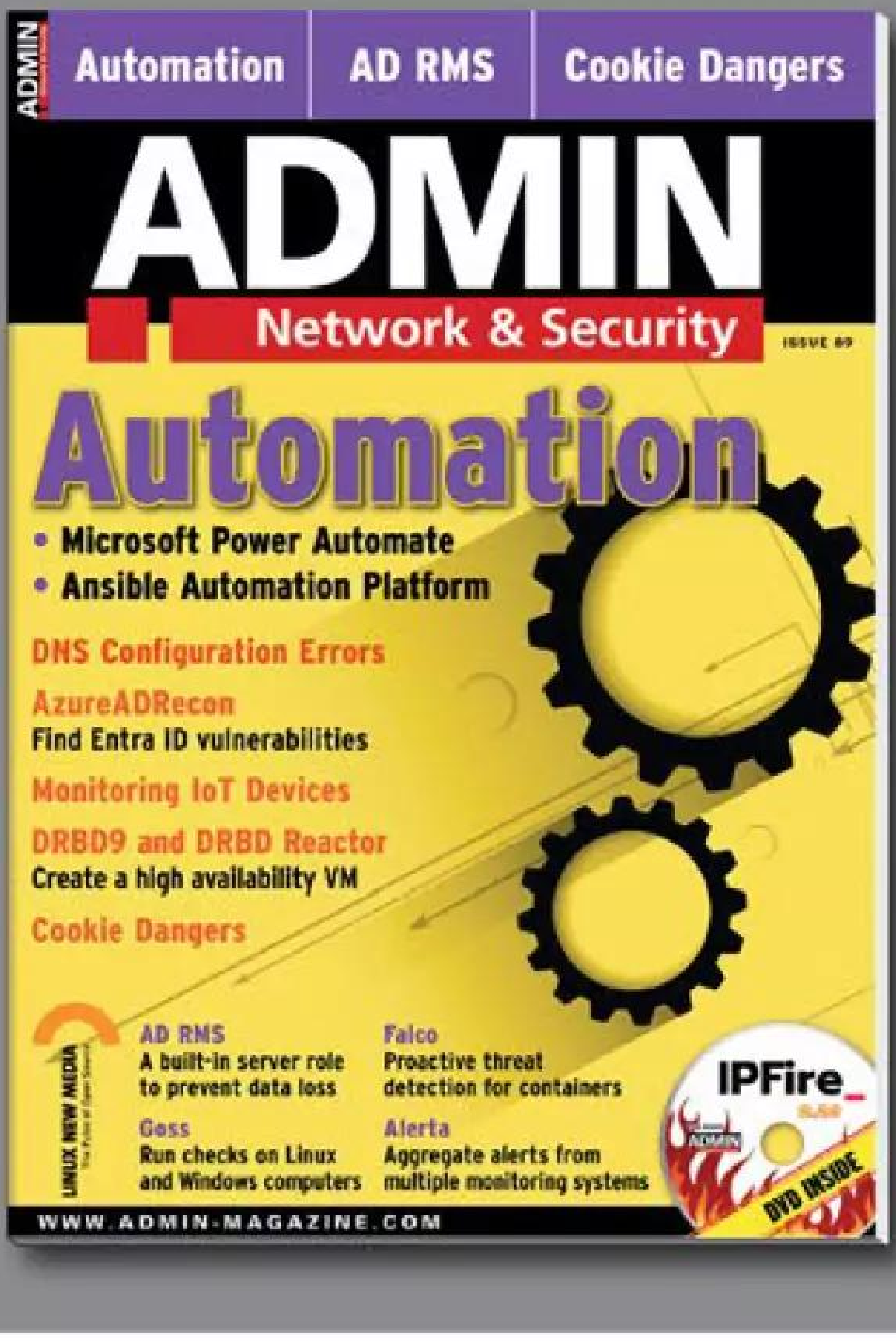
Want to subscribe?

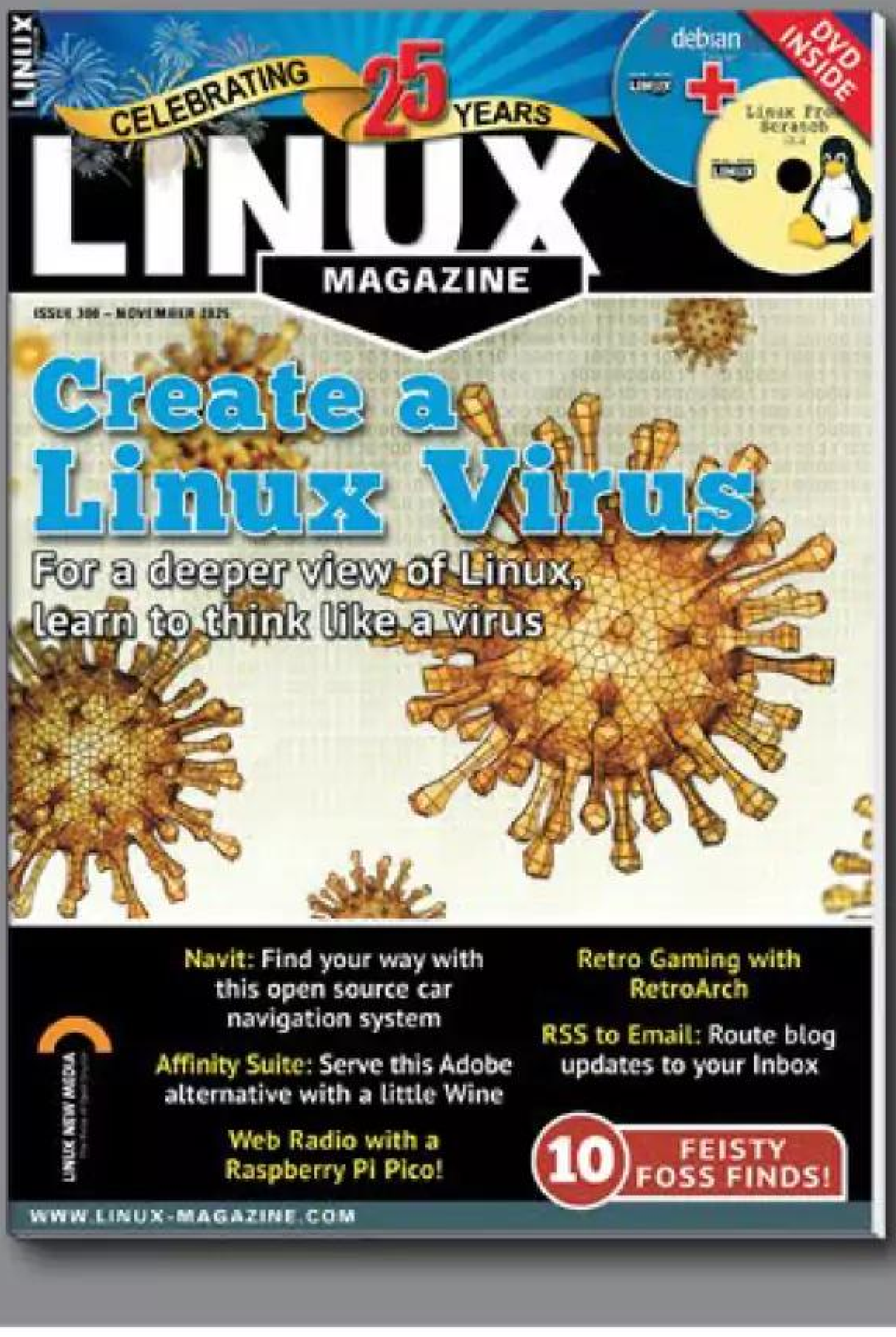
Searching for that back issue you really wish you'd picked up at the newsstand?

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

▶▶ shop.linuxnewmedia.com/shop ◀◀

DIGITAL & PRINT SUBSCRIPTIONS


ADMIN
Network & Security
Automation
AD RMS
Cookie Dangers
WWW.ADMIN-MAGAZINE.COM


LINUX MAGAZINE
Celebrating 25 Years
Create a Linux Virus
WWW.LINUX-MAGAZINE.COM

SPECIAL EDITIONS

Getting Started with Linux
LibreOffice Expert
Linux Shell Handbook
Cool Linux Hacks
MakerSpace #05
The Best of Linux Magazine 2024

Photo by Marija Zaric on Unsplash

Hotpatching in Windows Server 2025

Trouble-Free

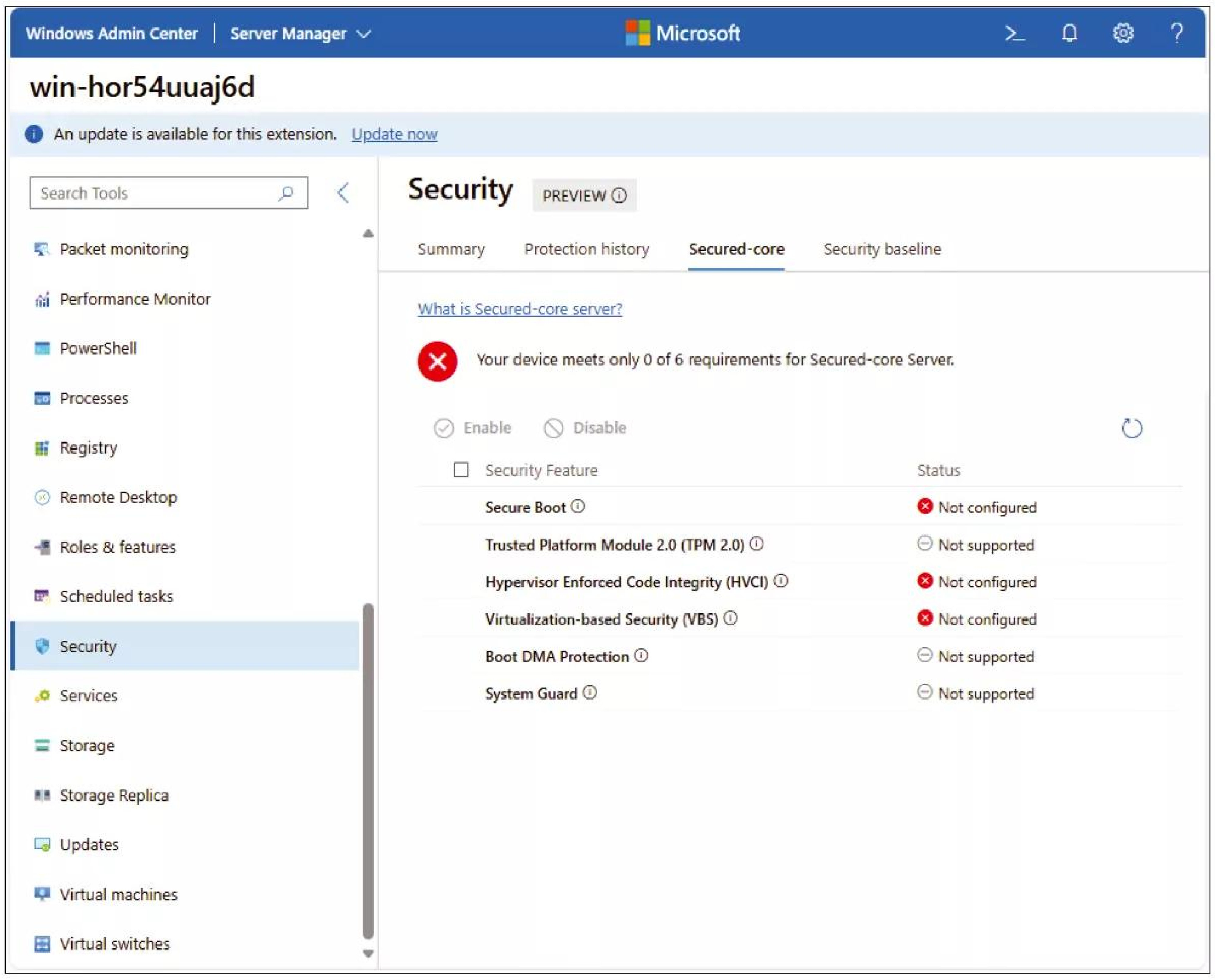
Hotpatching technology on Windows Server 2025 reduces downtime and maintenance windows, but it comes at a price in the truest sense of the word. We look at the technical details of hotpatching, show how to get started, and weigh the pros and cons. By Thomas Joos

Hotpatching directly modifies the code of processes running in RAM, eliminating the need for a restart. The key advantage is reduction of downtime, as servers remain operational and security updates are made without delay. Not all updates support this technology, though, and hotpatching will probably be limited to regular patch day updates in the future. Because not all updates support hotpatching, rebooting is still necessary, which certainly puts the benefits into perspective. In this article, you'll also learn about more limitations.

Microsoft Azure Required

On Windows Server 2022, you need the *Datacenter: Azure Edition* license, which must be running in Azure or Azure Stack hyperconverged infrastructure (HCI), to use hotpatching. Although hotpatching will also be available in local data centers in the Standard and Datacenter editions in the future, you still need an Azure connection.

To use hotpatching, you must first enable virtualization-based security (VBS) on Windows Server 2025 in the *Security | Secured-core* section of the Windows Admin Center (Figure 1). VBS uses Hyper-V virtualization technology to move security-critical processes to an isolated environment. VBS isolates particularly sensitive data and processes from the regular operating system



Windows Admin Center | Server Manager
Microsoft
win-hor54uuaj6d
Update now
Search Tools
Packet monitoring
Performance Monitor
PowerShell
Processes
Registry
Remote Desktop
Roles & features
Scheduled tasks
Security
Services
Storage
Storage Replica
Updates
Virtual machines
Virtual switches
Security
PREVIEW
Summary
Protection history
Secured-core
Security baseline
What is Secured-core server?
Enable
Disable

An update is available for this extension.

Your device meets only 0 of 6 requirements for Secured-core Server.

Security Feature	Status
Secure Boot	Not configured
Trusted Platform Module 2.0 (TPM 2.0)	Not supported
Hypervisor Enforced Code Integrity (HVCI)	Not configured
Virtualization-based Security (VBS)	Not configured
Boot DMA Protection	Not supported
System Guard	Not supported

Figure 1: The hotpatch feature in Windows Server 2025 requires virtualization-based security.

environment, which greatly limits the possibilities for attack.

By combining VBS with features such as Credential Guard, Windows Server 2025 protects access credentials by shifting their processing to a virtualized environment, minimizing the risk of credential theft attacks. New in Server 2025 is enhanced integration of VBS in multitenant environments, which now enables improved isolation between different tenant data while optimizing performance. On top of this, Server 2025 offers greater flexibility in terms of the implementation, thanks to additional configuration options, and makes it easier for you to manage and monitor VBS instances.

The Azure portal is used to configure hotpatching in Windows Server 2025, rather than group policies or settings in Windows Update. You first need to work through the pre-installed wizard for Azure Arc on the server to connect the server to Azure.

Another limitation is that hotpatching is not free; you need an Azure subscription, which has additional costs for the hotpatching subscription. Only companies that use Azure Stack HCI or run Windows servers in Azure receive the technology free of charge. Once you have worked through the Azure Arc wizard (**Figure 2**), hotpatching can be enabled with just a few clicks in the server object within Azure. No settings or notification functions are available. Basically, you start hotpatching and then check Windows Update from time to time to see if updates have been installed in this way.

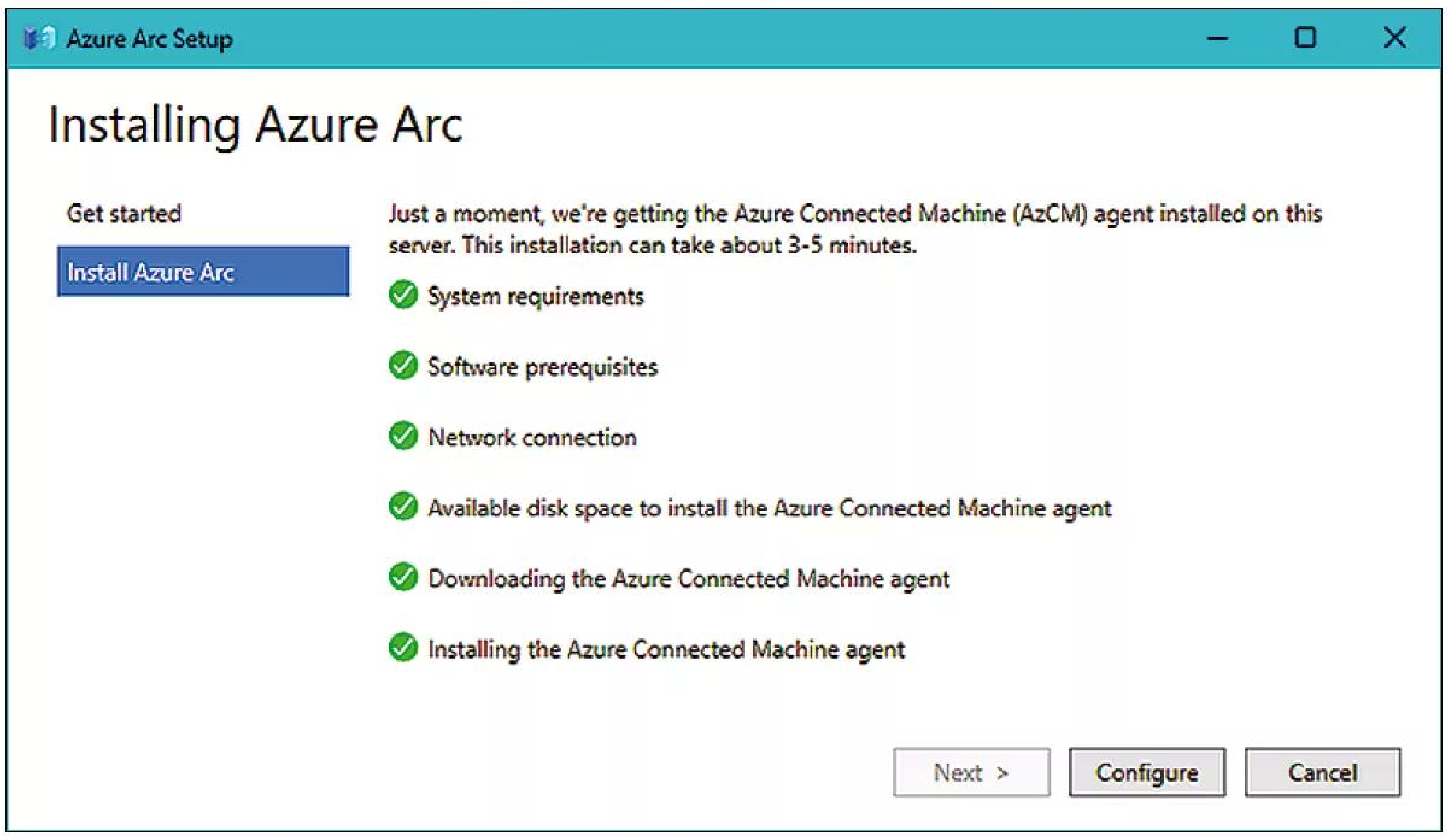
The feature is likely to be useful in high-availability clusters in larger environments, but less so in small and medium-sized businesses.

Architecture and Technical Implementation

The system loads the code sections that have been updated by hotpatching into a separate memory area without directly modifying the original files on disk. The operating system then redirects all calls of the old code to the new memory areas. The kernel monitors the affected memory areas and implements hooks to ensure that the new code integrates seamlessly. Before the hotpatch files are loaded, they are validated with cryptographic signatures, which ensures that only patches signed and verified by Microsoft are used.

Page table entries (PTEs) play a crucial role in hotpatching in Windows Server 2025 by enabling direct access to memory pages used by a patch. Hotpatching allows security-related updates to be installed without restarting the system by modifying RAM in a targeted way. PTEs control which memory areas are assigned to an application or process and map new code sections to memory without affecting the existing execution.

Instead of having to replace entire libraries, the revised architecture means that individual functions within a library can be patched. The benefit is a significant reduction in update size. Additionally, hotpatching supports security-critical kernel drivers that required a restart in previous versions. This extended kernel support improves the efficiency and flexibility of update processes. Memory optimization is another highlight: Old code segments remain in memory as long as dependencies exist and are only removed after a specified time. This approach minimizes the risk of crashes in running processes and improves integration with monitoring tools.



Azure Arc Setup

Installing Azure Arc

Just a moment, we're getting the Azure Connected Machine (AzCM) agent installed on this server. This installation can take about 3-5 minutes.

- System requirements
- Software prerequisites
- Network connection
- Available disk space to install the Azure Connected Machine agent
- Downloading the Azure Connected Machine agent
- Installing the Azure Connected Machine agent

Figure 2: Hotpatching requires a connection to Azure.

The typical workflow for a hotpatch update begins with deployment by the Windows Update system. After downloading, the server analyzes the patch, validates the signatures, and loads the new components into memory. The kernel then sets hooks that redirect all relevant system calls to the new code sections. An integrity check ensures that the changes have been applied correctly. Finally, Windows 2025 removes the old code segments from memory after a certain period of time, provided no dependencies remain.

Despite these advantages, hotpatching also has its limitations. In complex environments, dependencies between components can cause problems if multiple parts cannot be patched at the same time. Moreover, the hooks and memory redirects used can cause a minor performance overhead in some scenarios, which can translate to higher CPU and memory load. Furthermore, not all updates can be implemented as hotpatches. Changes to basic boot components still require a restart.

HotPatching When Restarting the Server

Windows permanently integrates the hotpatch updates, which are initially only active in RAM, into the operating system when the server restarts. The operating system synchronizes the updated components that were temporarily stored in memory with the matching files on disk, which means the patched state is retained after restarting.

After a restart, the system removes all the hooks and redirects configured for the hotpatches at runtime. Because the new versions of the components are available directly as primary executable files and no redirects are necessary, Windows no longer needs the hooks. For security-critical updates to the kernel or drivers that were patched during uptime but were not yet fully active, the system loads the new versions when restarting. Windows then initializes them directly as part of the boot process. If a hotpatch update is faulty or causes incompatibilities, restarting provides a route to returning the system to a stable state.

The advantages of hotpatching are, of course, fewer restarts and smaller update sizes. Maintenance windows are shorter and updates can be installed more quickly. During hotpatching, Windows does not interrupt any services; they just continue to run in the background despite the installation. The more servers you have in use, the more worthwhile hotpatching is likely to be in terms of total update time reductions. Large networks will therefore benefit disproportionately from the new feature.

Azure Arc as the Basis for Hotpatching

In Windows Server 2025, the icon for setting up Azure Arc is available in the taskbar tray area. This process installs the Azure Connected Machine agent on the server, after which the connection is set up by clicking *Configure*. Once communication between the local server and Azure is established, the server can be managed by the Windows Admin Center on the Azure portal, just as at the local datacenter. A VPN is not required; the connection uses the agent that was just installed.

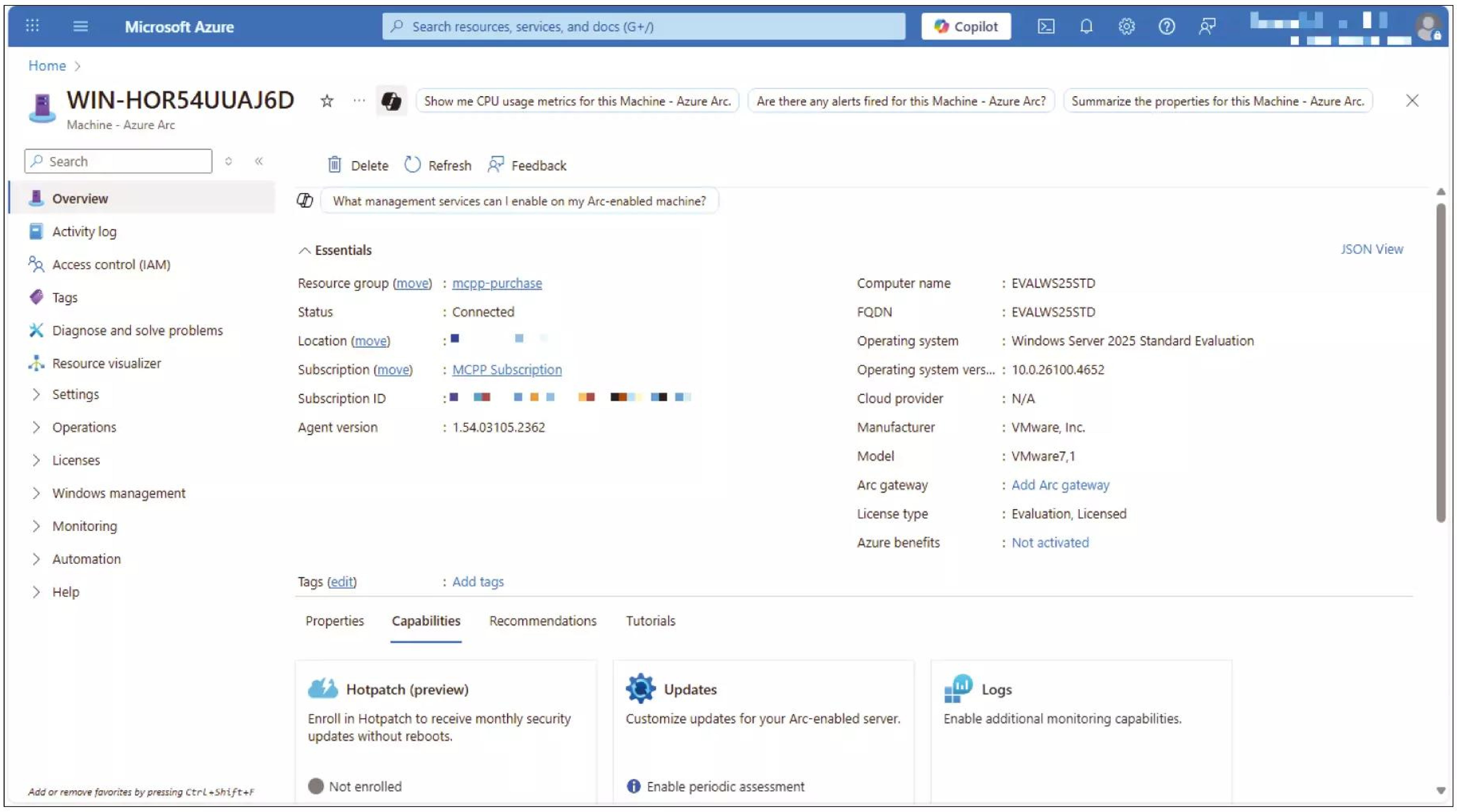
Azure Arc is a service in Azure that lets you connect local servers to Azure by an agent. Doing so makes it possible to secure, manage, monitor, and control servers from a web browser and the Azure portal from wherever you are, just as you would on a local network (Figure 3). Special services such as hotpatching can only be enabled on the Azure portal after connecting to Azure Arc. The connection between local servers and Azure with the Azure Connected Machine agent installed on the server also eliminates the need for a virtual private network (VPN). At the same time, you can access the server from wherever you happen to be and connect it to services in Azure or manage and monitor it remotely.

In addition to individual servers connected with the agent, you can also connect servers from Update Management to Azure Arc under *Azure Arc | Server* on the Azure portal for centralized management. To connect multiple machines from a datacenter to Azure Arc in one fell swoop, you can use a script that you generate by selecting *Add multiple servers | Generate script*. In the window that appears, you then enter the data for your servers and download the script.

Connecting to Azure Arc offers additional benefits: Linking local servers to the cloud means organizations can benefit from Azure services in their local datacenter. Examples include provisioning managed SQL instances, PostgreSQL, web apps, or low-code applications, making it easy to scale IT environments without having to subscribe to extensive cloud infrastructures or migrate data from the local datacenter to the cloud.

Enabling Hotpatching

In the Update Manager, the Windows Server 2025 user interface shows whether hotpatching was


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
[redacted]
Home
WIN-HOR54UUAJ6D
Machine - Azure Arc
Show me CPU usage metrics for this Machine - Azure Arc.
Are there any alerts fired for this Machine - Azure Arc?
Summarize the properties for this Machine - Azure Arc.
Search
Delete
Refresh
Feedback
What management services can I enable on my Arc-enabled machine?
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Operations
Licenses
Windows management
Monitoring
Automation
Help
Add or remove favorites by pressing Ctrl+Shift+F
Essentials
JSON View
Properties
Capabilities
Recommendations
Tutorials

Resource group (move) : mcpp-purchase
Status : Connected
Location (move) : [redacted]
Subscription (move) : MCPP Subscription
Subscription ID : [redacted]
Agent version : 1.54.03105.2362
Tags (edit) : Add tags

Computer name : EVALWS25STD
FQDN : EVALWS25STD
Operating system : Windows Server 2025 Standard Evaluation
Operating system vers... : 10.0.26100.4652
Cloud provider : N/A
Manufacturer : VMware, Inc.
Model : VMware7,1
Arc gateway : Add Arc gateway
License type : Evaluation, Licensed
Azure benefits : Not activated

Hotpatch (preview)
Enroll in Hotpatch to receive monthly security updates without reboots.
Not enrolled

Updates
Customize updates for your Arc-enabled server.
Enable periodic assessment

Logs
Enable additional monitoring capabilities.

Figure 3: Once a server is connected to Azure via Azure Arc, it appears as an object on the Azure portal.

used to install updates. Besides this, the *Hotpatch capable* status for updates tells you that hotpatching support is available. As part of the Azure Arc connection configuration, you are logged in to the Azure portal. The connection is then completed and the server is connected to the Azure portal.

If everything works out, the wizard will let you know, and your server will now be visible on the Azure portal. At the bottom of the window, you can now set up the feature with just a few clicks. As explained earlier, virtualization-based security is required on the server for hotpatching. You need to check now whether VBS is active and configured. You can do this in the Windows Admin Center after connecting to the server with *Security* | *Secured-core*. If VBS is active, the server can be enabled for hotpatching by confirming the option *I want to license this Windows Server to receive monthly hotpatches* (Figure 4). This process takes between five and 10 minutes, after which the status is visible in the window. The Azure portal then displays an *Enabled* status for the server in the *Hotpatch* section. You can disable hotpatching at any time in the settings.

The cost of hotpatching depends on your licensing setup. However, each hotpatch-enabled server running Windows Server 2025 requires an additional subscription to this feature. Licensing is separate from the server itself and is provided as a separate service and subscription. You can also enable hotpatching for individual servers only, which allows special servers to benefit from the technology while other machines continue to install updates in the traditional way. The price is based on Azure Update Manager and is $5 per server and month.

Windows 11 Also to Receive Hotpatching

In the future, Windows 11 will also have a hotpatching feature [1], which means the operating system can update autonomously without rebooting. However, you need to be aware of some limitations. The technology integrates updates directly into the RAM of active processes, just as on Server 2025. In Windows 11, this will probably mainly be relevant for the cumulative updates that are released on Patch Day each month. Microsoft expects this to reduce the number of mandatory restarts from 12 to four. It's not the case that Windows 11 will never need to reboot again, and only Windows 11 Enterprise will have hotpatching, but not the other editions. What is currently unclear is whether Windows 11 Pro will also benefit from these advantages in the long term.

Linux Magazine Subscription
Print and digital options
12 issues per year

SUBSCRIBE
shop.linuxnewmedia.com/shop

Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- News on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!

Need more Linux?
Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

bit.ly/Linux-Update

Additionally, the computer must be connected to Azure, which requires a subscription to Windows 11 Enterprise E3 and E5 (Microsoft 365 A3 and A5 or Microsoft 365 F3) or one of Windows 365. The computers need to have at least Windows 11 version 24H2 installed and be managed by Microsoft Intune.

Finally, it is important to note that hotpatching follows an annual cycle. In the first month of each calendar quarter (January, April, July, and October), devices receive the regular cumulative security update, followed by a restart. These updates include security fixes as well as new features and enhancements. In the following two months, servers receive hotpatch updates that contain only security-related updates. These are applied directly without requiring a restart. The cycle starts again at the beginning of the next quarter.

Conclusions

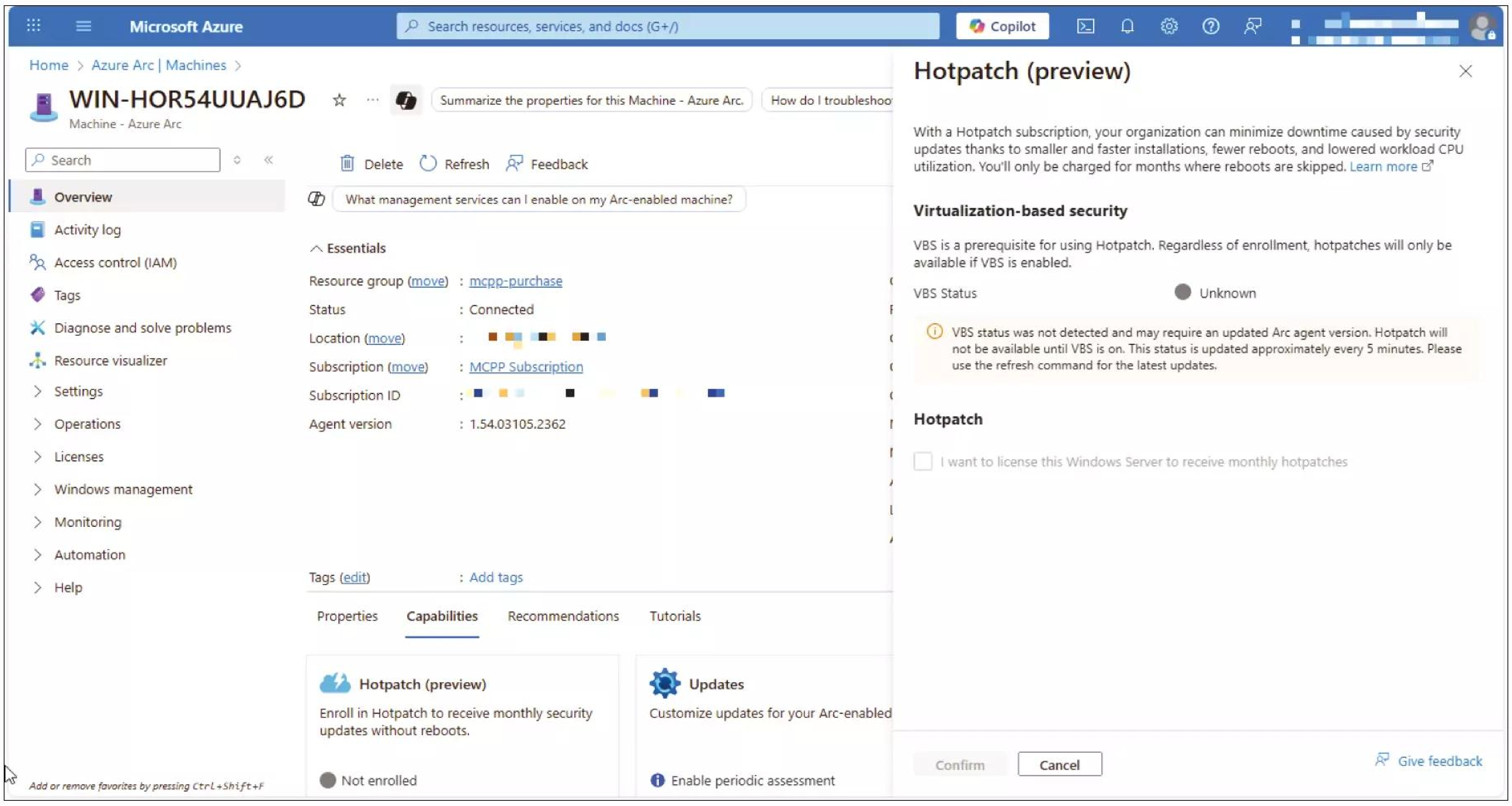
Hotpatching in Windows Server 2025 offers a way to implement security-critical updates without restarting and to reduce downtime in the process. The technology opens up new possibilities for optimizing maintenance processes, especially for high-availability environments and larger networks. However, the benefits are offset by limitations such as limited update support and the need for a paid Azure subscription. For organizations that rely on Azure infrastructures, hotpatching is a valuable addition, whereas admins in smaller environments will need to weigh carefully whether the benefits justify the additional costs. Ultimately, hotpatching remains a step toward more efficient system maintenance, but it does not completely eliminate all existing patch day challenges. ■

Info

[1] Windows 11 hotpatching: [https://techcommunity.microsoft.com/blog/windows-itpro-blog/hotpatch-for-client-comes-to-windows-11-enterprise/4302717]

The Author

Thomas Joos is a freelance IT consultant and has been working in IT for more than 20 years. In addition, he writes hands-on books and papers on Windows and other Microsoft topics. Online you can meet him on [http://thomasjoos.spaces.live.com].


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
[redacted]
Home > Azure Arc | Machines >
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Operations
Licenses
Windows management
Monitoring
Automation
Help
Add or remove favorites by pressing Ctrl+Shift+F
Summarize the properties for this Machine - Azure Arc.
How do I troubleshoo
Delete
Refresh
Feedback
What management services can I enable on my Arc-enabled machine?
Essentials
Properties
Capabilities
Recommendations
Tutorials
Confirm
Cancel
Give feedback

WIN-HOR54UUAJ6D

Machine - Azure Arc

Resource group (move) : mcpp-purchase

Status : Connected

Location (move) : [redacted]

Subscription (move) : MCPP Subscription

Subscription ID : [redacted]

Agent version : 1.54.03105.2362

Tags (edit) : Add tags

Hotpatch (preview)

Enroll in Hotpatch to receive monthly security updates without reboots.

● Not enrolled

Updates

Customize updates for your Arc-enabled

Enable periodic assessment

Hotpatch (preview)

With a Hotpatch subscription, your organization can minimize downtime caused by security updates thanks to smaller and faster installations, fewer reboots, and lowered workload CPU utilization. You'll only be charged for months where reboots are skipped. Learn more

Virtualization-based security

VBS is a prerequisite for using Hotpatch. Regardless of enrollment, hotpatches will only be available if VBS is enabled.

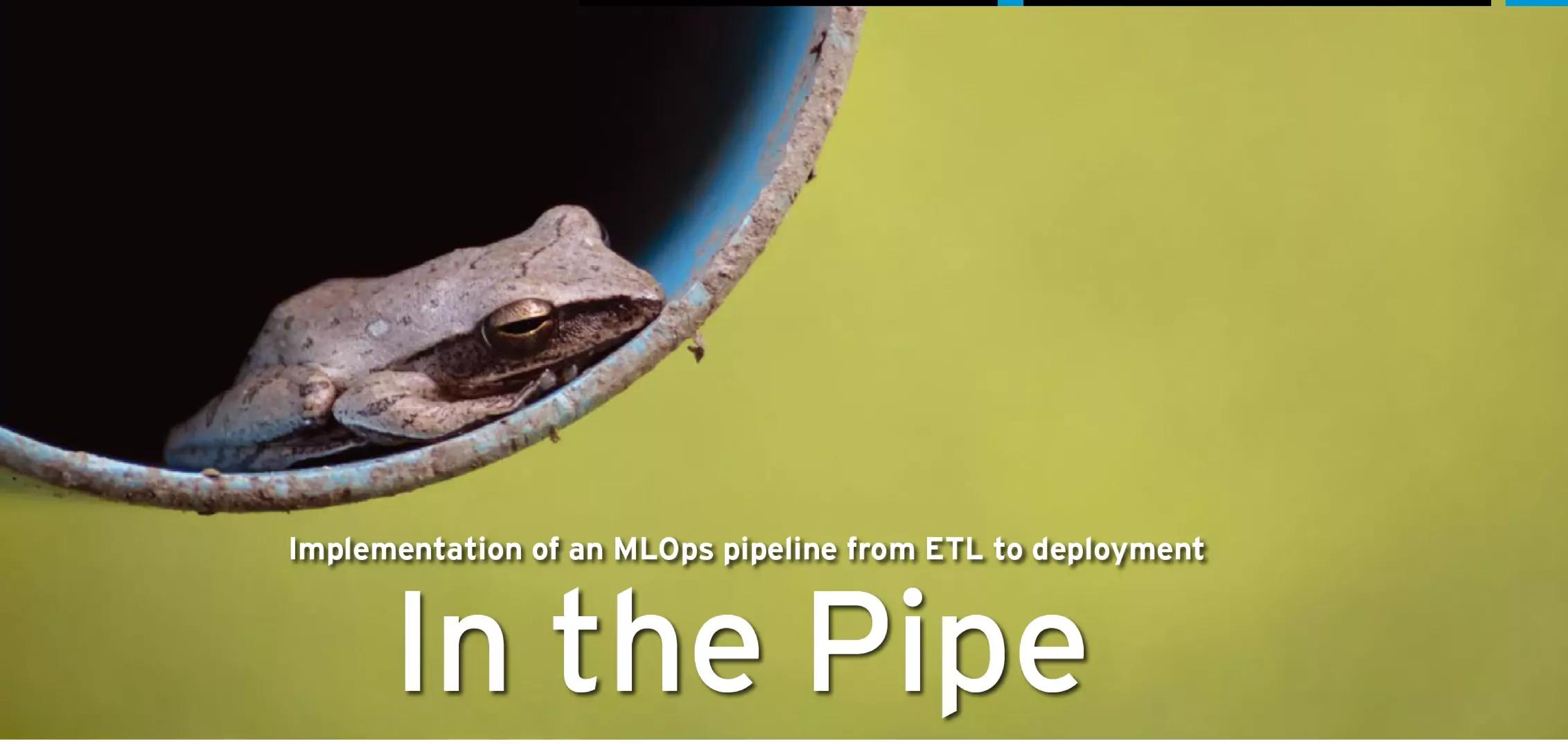
VBS Status ● Unknown

VBS status was not detected and may require an updated Arc agent version. Hotpatch will not be available until VBS is on. This status is updated approximately every 5 minutes. Please use the refresh command for the latest updates.

Hotpatch

☐ I want to license this Windows Server to receive monthly hotpatches

Figure 4: Once Azure Arc and Windows Server 2025 are set up for hotpatching, all you need to do is check the box for the commercial subscription.



Implementation of an MLOps pipeline from ETL to deployment

In the Pipe

We present a production-grade MLOps pipeline that automates the entire machine learning lifecycle, from data ingestion to deployment in a Kubernetes environment. By Tanish Pathania, Dr. B. Thangaraju, and Siddharth Vikram

We deliver a production-grade machine learning operations (MLOps) [1] pipeline that accommodates the entire machine learning (ML) application lifecycle from ingestion of data to deployment in near-real time in a Kubernetes context. The pipeline has three primary phases of operation: (1) A mature extract, transform, and load (ETL) process reads in semi-structured and structured data, cleanses and transforms it, and saves the output in a scalable MongoDB instance; (2) dynamic model training reads in fresh data from MongoDB and employs reproducible, modulated scripts to achieve on-demand, decoupled computation; and (3) all parts are `dockerized` and shipped to Minikube hosting in a local cluster of Kubernetes.

Deployment and service configurations reveal near-real time inference by `NodePort` services. The system follows current DevOps techniques of modulated deployment, compositionality, and scaling and forms the basis of realizing in the near future additional extensions (e.g., continuous integration and continuous delivery (CI/CD) and observability) and deployment to the cloud, resulting in a working, cloud-agnostic method of deployment and scaling of ML systems.

ML Deployment Difficulties

Machine learning deployment in production is usually hindered by disjointed workflows, manual intervention, and a pipeline that is not automated end-to-end. Conventional ML pipelines – covering data ingestion through deployment – are normally disjointed, script-based, and not very integrated with contemporary DevOps methods. This disjointedness creates major hurdles in scaling, reproducibility, and maintainability and inhibits the operationalization of ML models in applications.

A number of systemic problems lie behind this inefficiency. Pipelines are often constructed as discrete pieces that interact with each other in places where friction arises during multiple staged transitions. Repetitive and manual work is typical of retraining and deployment tasks with little to no automation. Lack of systematic version control makes reproducing results or auditing model activity on deployment even more challenging. These deficiencies worsen as datasets increase and the size and scope of model architectures expand so that the limitations of the normal ML workflow in performance, observability, and operational burden stand revealed. Moreover, conventional ML systems often lack seamless integration with DevOps tooling.

Unlike software development pipelines, ML workflows involve non-deterministic outputs, evolving data dependencies, and iterative experimentation – all of which require specialized infrastructure to manage effectively. Without robust CI/CD integration, model development and deployment remain siloed, delaying delivery cycles and increasing technical debt. As a response to these constraints, we present a unified and modular pipeline of MLOps that automatically manages the entire machine learning life cycle. This pipeline is organized into three main components (Figure 1). First, an ETL pipeline is used to retrieve and transform raw data and store it in a MongoDB database. Second, a dynamic training module fetches the same data during runtime to train the ML algorithms, without depending on a persistent intermediate state. Third, the pipeline makes use of Docker to dockerize the training and inference workflow with Kubernetes – provisioned locally with Minikube-handling deployment and orchestration of services.

By combining containerization, orchestration, and scale-out storage with DevOps best practices, this MLOps workflow meets the key shortcomings of existing ML systems. The resulting pipelines guarantee reproducibility and ease of deployment and set the stage for hassle-free CI/CD compatibility, making it a promising base for production-ready ML deployments.

Lead Image © jame505th, 123RF.com

Related Work

A number of open source tools have driven MLOps best practices (Figure 2). Prominent among these are MLflow [2], Kubeflow [3], and TensorFlow Extended (TFX) [4]. MLflow eases experiment tracking, model tracking, and automated deployment. Kubeflow, built to be compatible with Kubernetes, offers an end-to-end orchestration and monitoring solution for ML pipelines. TFX, built by Google, provides end-to-end automated ML pipelines [5], with TensorFlow-based environments in mind.

In the industry, strong examples at scale can be observed in large-scale systems, such as Uber's Michelangelo [6] and Google's TFX pipeline [4]. Michelangelo unifies model training, deployment, and monitoring and is specifically designed to suit Uber's high-demand requirements. The Google TFX pipeline, although TensorFlow-centric, offers strong, scalable ML workflow capabilities suitable in production environments. These platforms do well with individual aspects of the ML lifecycle, although they can be complicated to configure and do not always provide flexibility to use anything other than TensorFlow. Additionally, they tend to be geared toward scalable environments and therefore are less suited to small teams or low-overhead environments.

An enhancement of those principles is applied to offer an integrated, modular pipeline that incorporates all of the ML stages, ranging from ETL to deployment, and thus promoting flexibility and scalability and making setup simpler.

Methodology

The ETL pipeline [7] of this project comprises a number of individual components, each performing some particular task as part of the data processing pipeline. The orchestration of these components is controlled by `etl_pipeline.py`, and individual scripts handle particular parts of the pipeline.

1. Data Extraction. Data extraction from raw CSV files in the `data/raw/` directory is handled by the `extract.py` module (Listing 1). The main pipeline orchestrator starts the data extraction process and makes sure that raw data is loaded into a format that can be transformed. Each CSV file is parsed, processed, and stored in memory for later steps by the extraction script.

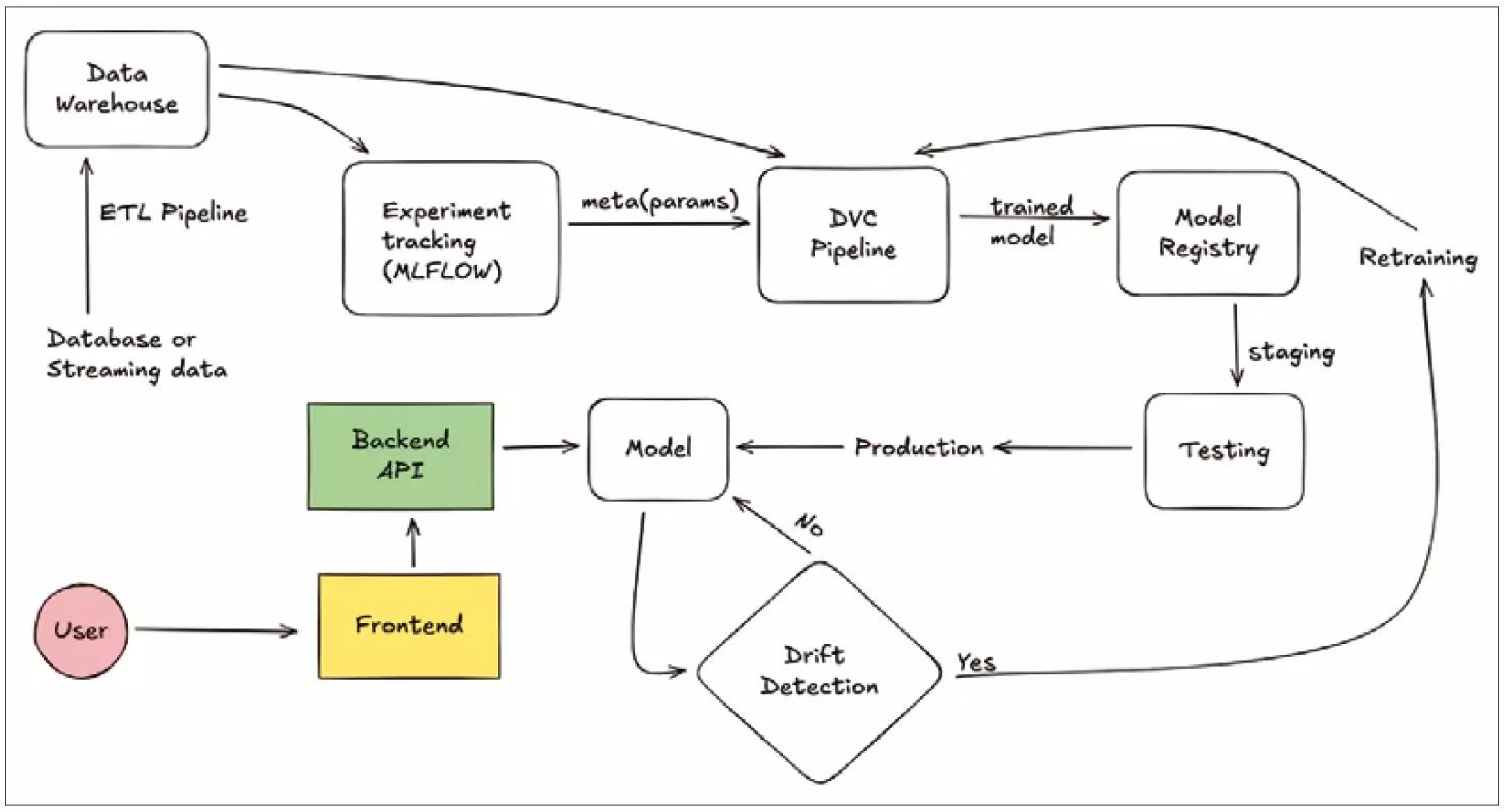
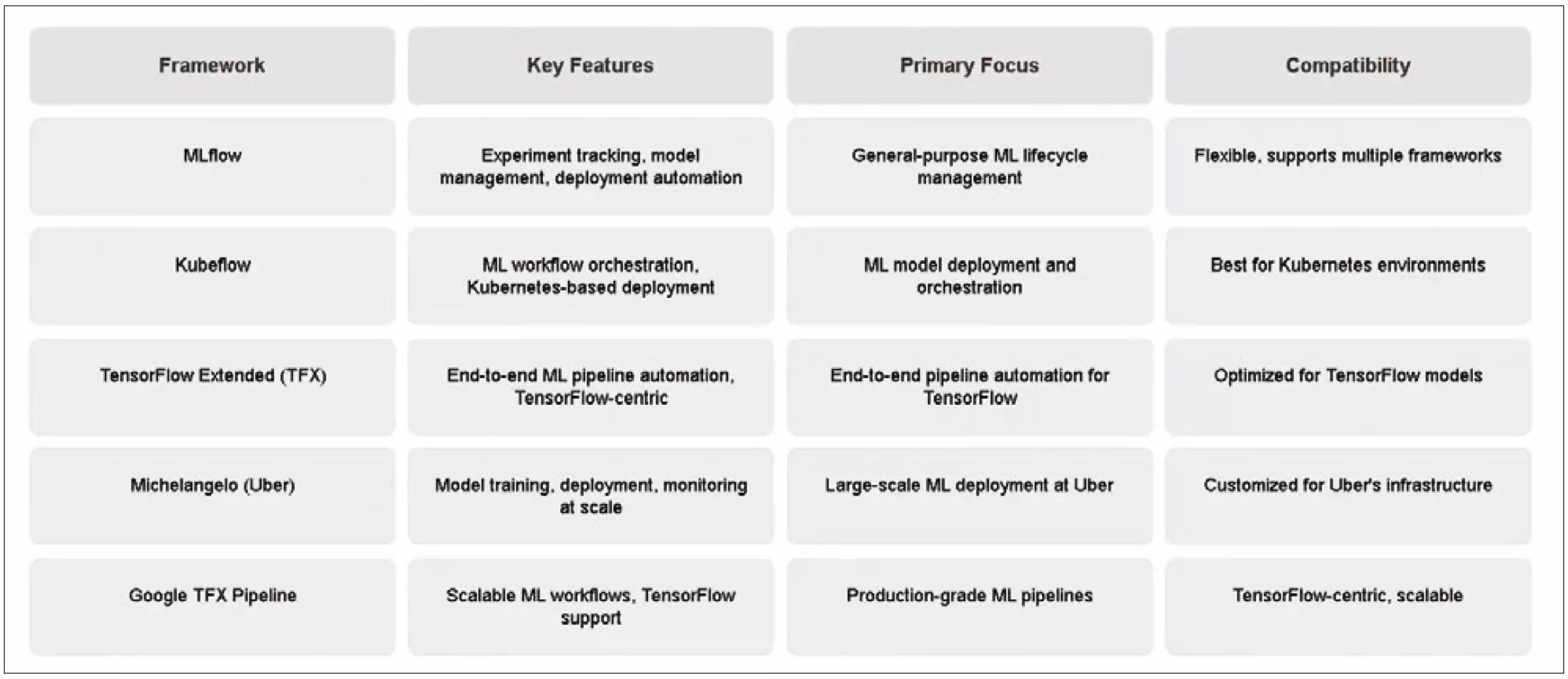


Figure 1: Flowchart of an MLOps pipeline.



Framework	Key Features	Primary Focus	Compatibility
MLflow	Experiment tracking, model management, deployment automation	General-purpose ML lifecycle management	Flexible, supports multiple frameworks
Kubeflow	ML workflow orchestration, Kubernetes-based deployment	ML model deployment and orchestration	Best for Kubernetes environments
TensorFlow Extended (TFX)	End-to-end ML pipeline automation, TensorFlow-centric	End-to-end pipeline automation for TensorFlow	Optimized for TensorFlow models
Michelangelo (Uber)	Model training, deployment, monitoring at scale	Large-scale ML deployment at Uber	Customized for Uber's infrastructure
Google TFX Pipeline	Scalable ML workflows, TensorFlow support	Production-grade ML pipelines	TensorFlow-centric, scalable

Figure 2: Comparison of open source MLOps frameworks.

2. Data Transformation and Cleaning. Handling missing data is among the crucial data cleaning and preprocessing tasks carried out by the `transform.py` module (Listing 2) to get the data ready for database loading and model training, including (a) dealing with missing values, wherein the module either removes the corresponding rows or columns or imputes them; (b) choosing pertinent features for model training; and (c) normalizing and scaling to guarantee the standardization of data features, particularly if the model calls for input within a consistent range.

3. Data Loading to MongoDB. Several essential functions are integrated by the `load.py` module: logging, which makes use of the module to make sure that every pipeline step is recorded for monitoring and debugging; and data validation, for which the `validate.py` module verifies the consistency and quality of the data before loading it into MongoDB, after which the `database.py` module dynamically inputs the cleaned data into a MongoDB database, abstracting the connection logic and schema design. The schema is made to allow for flexible querying for model training and further analysis, and the MongoDB database is organized to manage the data effectively.

4. Pipeline Orchestration. The `etl_pipeline.py` script (Listing 3) orchestrates the entire ETL pipeline. By combining the loading, extraction, and transformation processes, this script makes sure that data moves smoothly from one step to the next. It is in charge of controlling the pipeline's overall execution and invoking the relevant functions in each module.

Makefile Setup and Execution

A Makefile is used to control the pipeline's execution, automating the setup, execution, and cleanup procedures. Several significant targets are included in the Makefile:

- `run`: Completes the ETL pipeline by invoking the main pipeline script (`etl_pipeline.py`), executing the extraction and validation processes, and setting up the required directories.
- `install`: Installs all necessary dependencies as listed in `requirements.txt`.
- `clean`: Eliminates logs and temporary files produced while the pipeline was running and verifies that the extracted data satisfies quality standards by performing a data validation check before processing starts.
- `check-env`: Confirms that the environment is set up correctly by ensuring that important environment variables like `MONGO_URI` are set and that required files like `.env` are present.

Table 1 shows how components are arranged in the project directory. When combined, these files provide a robust and adaptable ETL pipeline that can readily grow to accommodate more data sources or transformations.

Model Training

This project element exemplifies contemporary ML engineering. Its sturdy, modular architecture allows it to coordinate the training process seamlessly and automatically while dynamically retrieving data from MongoDB. The architecture makes use of MLflow's sophisticated experiment tracking to guarantee that each training cycle is auditable and repeatable. The dynamic data retrieval mechanism ensures that the model uses the most recent dataset available at runtime by removing reliance on static storage.

By creating a robust connection to MongoDB with a standardized configuration supplied in `config.py`, the `data_warehouse_connect.py` module plays a crucial role. With each execution, this design guarantees that the

Table 1: Project Components

Module	Function
`config.py`	Manages configuration details specific to the environment.
`database.py`	Oversees MongoDB interactions and schema design.
`logger.py`	Offers logging functionality for monitoring pipeline execution.
`extract.py`	Oversees the process of extracting raw data from CSV files.
`transform.py`	Implements preprocessing and data cleaning procedures.
`validate.py`	Verifies the quality of the data before it is loaded into the database.
`etl_pipeline.py`	Coordinates the complete ETL procedure.

Listing 1: `extract.py`

```
import pandas as pd
import os
from logger import get_logger
logger = get_logger()
def extract_data(data_folder="../data/raw"):
dataframes = {}
try:
  for file in os.listdir(data_folder):
    if file.endswith(".csv"):
      file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)
    dataframes[file[:-4]] = df
    logger.info(f"Extracted {len(df)} records from {file}.")
except Exception as e:
  logger.error(f"Error extracting data: {e}")
return dataframes
```

Listing 2: `transform.py`

```
import pandas as pd
from logger import get_logger

logger = get_logger()
def transform_data(dataframes):
  for name, df in dataframes.items():
    for column in df.columns:
      if df[column].isnull().any():
        if df[column].dtype == 'object':
          df[column] = df[column].fillna(df[column].mode()[0])
        else:
          df[column] = df[column].fillna(df[column].mean())
    logger.info(f"Transformed data for {name}: filled missing values.")
  return dataframes
```

Listing 3: `etl_pipeline.py`

```
from transform import transform_data
from load import load_data_to_mongo
from logger import get_logger
logger = get_logger()
if name == "main":
  logger.info("Starting ETL Pipeline...")
  dataframes = extract_data()
  transformed_data = transform_data(dataframes)
  load_data_to_mongo(transformed_data)
  logger.info("ETL Pipeline completed successfully!")
```

pipeline retrieves the most recent data. With this dynamic connection, the `preprocess.py` script performs extensive data cleaning, feature engineering, and normalization, methodically preparing the dataset for the best possible training results. By using the most recent database entries, this procedure guarantees that the data is cleaned and ready for training.

The `train.py` script, which is in charge of training a logistic regression model on the preprocessed data, is the central component of the model training pipeline, logging metrics, models, hyperparameters, and experiments with the use of MLflow. The following steps are part of the training pipeline:

- loading the target variable `y.csv` and the preprocessed feature set `X.csv`;
- dividing the data into sets for testing and training;
- scaling the features with `StandardScaler` to guarantee the logistic regression [8] model performs at its best;
- using the test set to train and assess the logistic regression model; and
- logging the accuracy score, model coefficients, and other training and model metrics to MLflow [2].

The core training logic is demonstrated in the snippet of `train.py` shown in Listing 4, which covers data loading, model training, evaluation, and logging by MLflow.

This pipeline guarantees that all artifacts are saved for future use and that the model is trained reproducibly. The Makefile is used to execute this orchestration, which simplifies the procedure and exhibits excellent modularity and repeatability. By encapsulating the end-to-end execution, the commands

```
make preprocess
make train
make all
```

strengthen the system's efficiency and integrity.

To serve predictions with `flask`, the `inference.py` script (Listing 5) offers a RESTful endpoint. It converts incoming JSON data, loads the scaler parameters and trained model coefficients, and generates binary loan approval predictions. This stateless microservice enables seamless integration with downstream applications and deployment pipelines.

In addition to automating the workflow's essential steps, the Makefile (Listing 6) guarantees that each stage can be independently tested and deployed by:

- minimizing the possibility of human error by reducing manual overhead;
- offering a scalable solution that can be incorporated into continuous delivery pipelines for CI/CD; and
- keeping a high level of flexibility to adjust to changing production needs.

The combination of dynamic data handling, thorough preprocessing, and efficient model training is embodied by this model training framework, which is more than just a set of scripts; it is a sophisticated, scalable solution. In the field of MLOps, it is a remarkable piece of work that exemplifies both operational effectiveness and technical excellence.

Listing 4: `train.py` (Snippet)

```
import pandas as pd, mlflow, mlflow.sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

mlflow.set_experiment("Loan_Approval_Experiment")

with mlflow.start_run():
  X = pd.read_csv("data/X.csv")
  y = pd.read_csv("data/y.csv")["Loan_Status"]
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
  scaler = StandardScaler()
  X_train = scaler.fit_transform(X_train)
  X_test = scaler.transform(X_test)
  model = LogisticRegression(max_iter=1000).fit(X_train, y_train)
  mlflow.log_metric("accuracy", accuracy_score(y_test, model.predict(X_test)))
  mlflow.sklearn.log_model(model, "logistic_regression_model")
```

Listing 5: `inference.py`

```
@app.route("/predict", methods=["POST"])
def predict():
  data = request.get_json()
  df = pd.DataFrame(data)
  df = df[coefficients.index]
  df = (df - scaler_mean) / scaler_var.pow(0.5)
  linear_output = df.mul(coefficients).sum(axis=1) + intercept
  predictions = (linear_output > 0).astype(int).tolist()
  return jsonify(predictions)
```

Listing 6: Makefile

```
all: preprocess train
preprocess: python src/preprocess.py
train: python src/train.py
inference: python src/inference.py
install: pip install -r requirements.txt
setup: mkdir -p logs data/raw model
clean: rm -rf logs/*.log pycache model data mlruns
```

Listing 7: Multistage Dockerfile

```
Stage 1: Preprocessing
FROM python:3.12-slim AS stage1
WORKDIR /app
COPY requirements/stage_1.txt ./
RUN pip install -r stage_1.txt
COPY src/ src/
RUN python src/preprocess.py # Generates cleaned data

Stage 2: Training
FROM python:3.12-slim AS stage2
COPY --from=stage1 /app/data/X.csv /app/data/y.csv ./data/
COPY requirements/stage_2.txt ./
RUN pip install -r stage_2.txt
RUN python src/train.py # Produces model/scaler

Stage 3: Inference
FROM python:3.12-slim AS stage3
COPY --from=stage2 /app/model/*.csv ./model/
COPY requirements/stage_3.txt ./
RUN pip install -r stage_3.txt
EXPOSE 5000
CMD ["flask", "run", "--host=0.0.0.0"] # Web service
```

Containerization and Deployment

Docker [9] and Kubernetes are used to containerize and deploy the entire workflow, which includes preprocessing, training, and inference, to operationalize the machine learning pipeline in a reproducible and scalable way. The deployment method with Minikube and the step-by-step containerization strategy are explained in this section.

Each stage of the ML lifecycle is divided into distinct, self-contained layers in a multistage Dockerfile [10] (Listing 7). With a lightweight Python base image, the first stage manages preprocessing. Dependencies are installed with a specific `stage_1.txt` requirements file, the source code is copied, and the preprocessing script is run to produce clean feature and label datasets.

To create a trained model and matching scaler parameters, the second stage imports the artifacts from the first stage, installs the training-specific dependencies mentioned in `stage_2.txt`, and runs `train.py`.

The third and final step involves `flask` to instantiate an inference server. This stage exposes the application on port 5000 and contains the trained artifacts from the training stage, as well as all runtime dependencies from `stage_3.txt`.

For every pipeline component, this design pattern guarantees clear dependency management, smaller image sizes, and concern separation. The `--from=stageX` directive passes artifacts between stages, preserving state across layers without requiring the persistence of intermediate containers.

To ensure safe and automated publishing, each stage-specific image is tagged and uploaded to Docker Hub. The build and run lifecycle of every component is coordinated locally by the `docker-compose.yml` file.

The `depends_on` constraints are used to run the services in a sequential manner, making sure that the preprocessing container is finished before training starts and that training is finished before inference starts. To transfer data and model artifacts between pipeline stages, shared host directories are mounted into the containers. Before cluster deployment, this setup facilitates end-to-end validation, local reproducibility, and smooth debugging.

The pipeline is installed on a local Kubernetes cluster that Minikube has provisioned for production orchestration and simulation. Declarative manifests located in the `k8s/` directory define the deployment (Listing 8). The `inference-deployment.yaml` file sources the image from Docker Hub and defines a Kubernetes deployment with a single replica of the inference container. To improve portability and cluster independence, ephemeral volumes are mounted by `emptyDir` to replicate shared storage in a cloud-native manner without depending on `hostPath` volumes.

Listing 8: Deployment Manifest

```
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 1
  template:
  spec:
    containers:
    - name: inference
      image: tanish688/experiment_tracking-inference:latest
      ports:
      - containerPort: 5000
      volumeMounts:
      - name: model-volume
        mountPath: /model
      - name: logs-volume
        mountPath: /logs
    volumes:
    - name: model-volume
      emptyDir: {}
    - name: logs-volume
      emptyDir: {}
```

The corresponding `inference-service.yaml` defines a `NodePort` service that maps the container's port 5000 to a node-level port 30007, enabling external access to the model inference endpoint (Listing 9).

In addition to encapsulating DevOps best practices like versioned artifact tracking and containerized delivery, this infrastructure uses microservices and infrastructure as code to validate real-world deployment strategies. For downstream integration, the Docker and Kubernetes stack offers modularity, scalability, and CI/CD readiness.

Results

Kubernetes deployment, image distribution, and container orchestration were used to confirm that the entire MLOps pipeline was executed successfully. The experimental procedure is described in this section, along with important findings from each phase.

Table 2: Docker Image Size Reduction

Optimization Stage	Image Size	Reduction
Initial (baseline)	3GB	-
After using volumes	1GB	66.7%
After multistage build	300MB	70% from 1GB
Total reduction	300MB	90% from 3GB

The containerized ML stages – preprocessing, training, and inference – were independently built and tested with the command

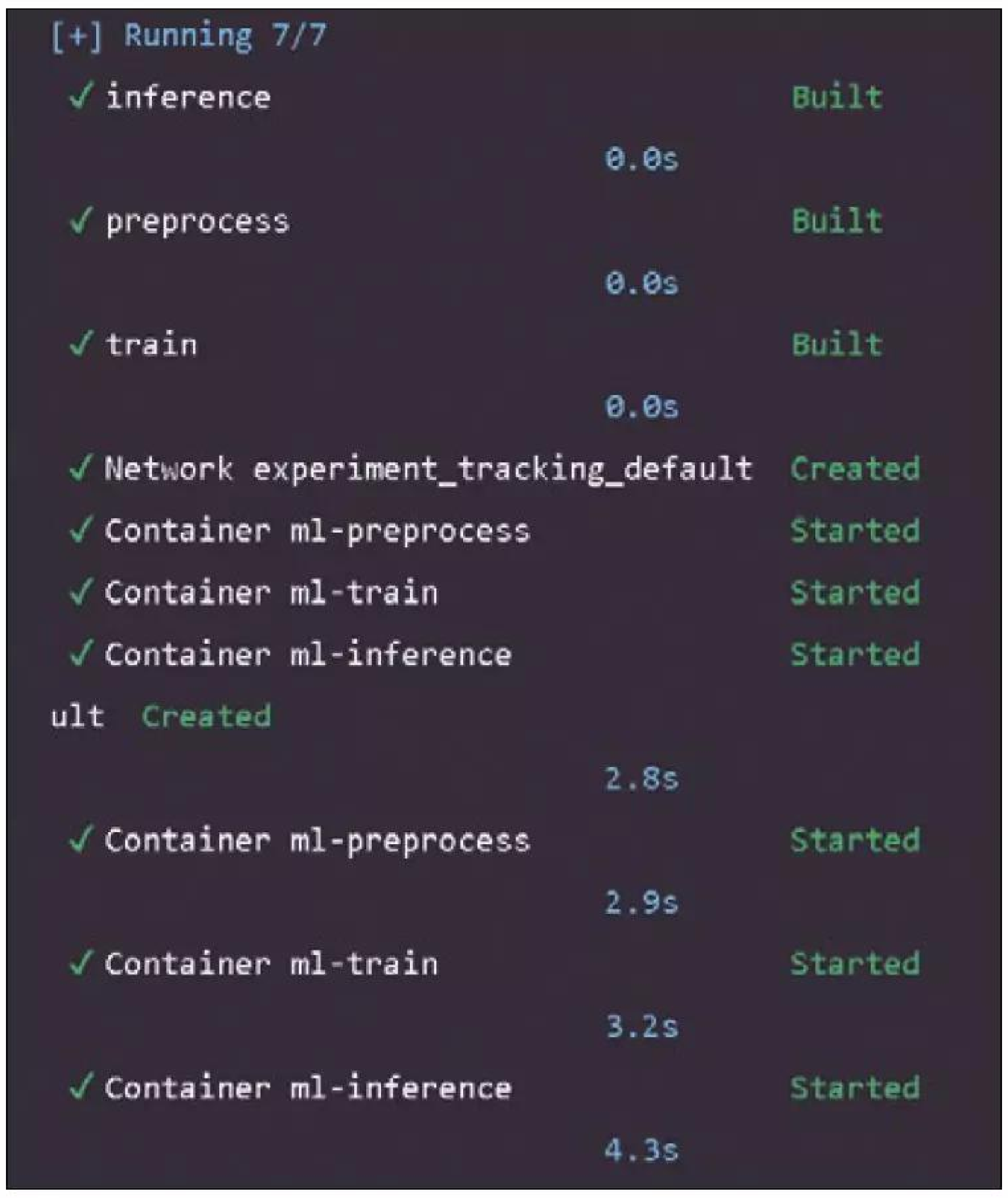
```
docker compose up --build -d
```

The detached-mode execution (`-d`) ensures all services run concurrently in isolated environments. Figure 3 displays the `compose` output, confirming successful builds and service execution.

Because of redundant files and bundled dependencies, the initial Docker images weighed in at about 3GB, so we used Docker volumes to externalize intermediate artifacts and data, which resulted in a 66.7 percent reduction in image size to 1GB (Table 2). We separated the build environment from the lightweight runtime image and used a multistage build approach to further optimize.

Listing 9: `inference-service.yaml`

```
apiVersion: v1
kind: Service
spec:
  type: NodePort
  ports:
  - port: 5000
    targetPort: 5000
    nodePort: 30007
  selector:
  app: inference
```



```
[+] Running 7/7
 ✓ inference                                   Built
                                    0.0s
 ✓ preprocess                                  Built
                                    0.0s
 ✓ train                                       Built
                                    0.0s
 ✓ Network experiment_tracking_default         Created
 ✓ Container ml-preprocess                     Started
 ✓ Container ml-train                          Started
 ✓ Container ml-inference                      Started
ult  Created
                                    2.8s
 ✓ Container ml-preprocess                     Started
                                    2.9s
 ✓ Container ml-train                          Started
                                    3.2s
 ✓ Container ml-inference                      Started
                                    4.3s
```

Figure 3: Docker Compose execution of all ML pipeline stages.

By eliminating superfluous build-time dependencies, the final image size was 300MB, which was 70 percent smaller than the 1GB intermediate image and 90 percent smaller than the initial 3GB baseline. Our Kubernetes-based infrastructure's deployment efficiency, container startup time, and resource usage were all greatly enhanced by these optimizations, which made use of `docker volume` [11].

The use of Docker Hub, a centralized registry for storing and sharing container images, made it easier to collaborate and deploy across various environments. We supported a CI/CD [12] workflow by facilitating smooth sharing between the development, testing, and production phases and pushing versioned images to Docker Hub (Figure 4).

Every pipeline image was successfully uploaded to Docker Hub by means of

```
docker push tanish688/experiment_↩
  tracking-preprocess:latest
docker push tanish688/experiment_↩
  tracking-train:latest
docker push tanish688/experiment_↩
  tracking-inference:latest
```

The inference service was deployed to a local Kubernetes cluster with the commands

```
kubectl apply-f k8s/inference-deployment.yaml
kubectl apply-f k8s/inference-service.yaml
```

Cluster state and logs were verified to ensure the pod was running and the inference service was correctly exposed by `NodePort`. Figure 5 presents a snapshot of the pod and service states, confirming successful deployment.

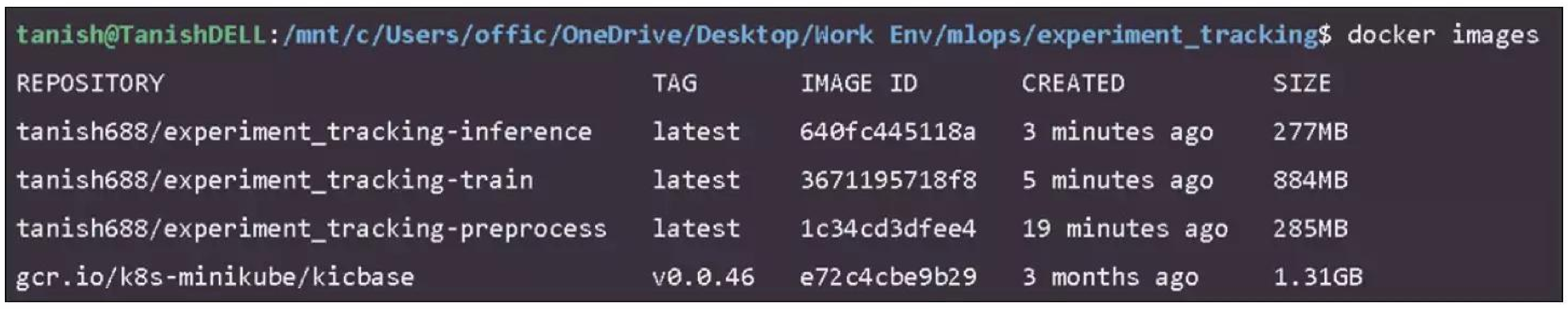
The ETL pipeline and experiment tracking components were executed more efficiently thanks to the automation made possible with the Makefile. By offering discoverable CLI operations, this interface improved repeatability and decreased manual errors.

The ETL pipeline's execution output is shown in Figure 6. Similarly, the help and result output for the experiment tracking pipeline are shown in Figures 7 and 8

A terminal-based test of the deployed inference API confirmed a full data-to-inference lifecycle. Figure 9 shows a successful request-response exchange between the client and the deployed model endpoint.

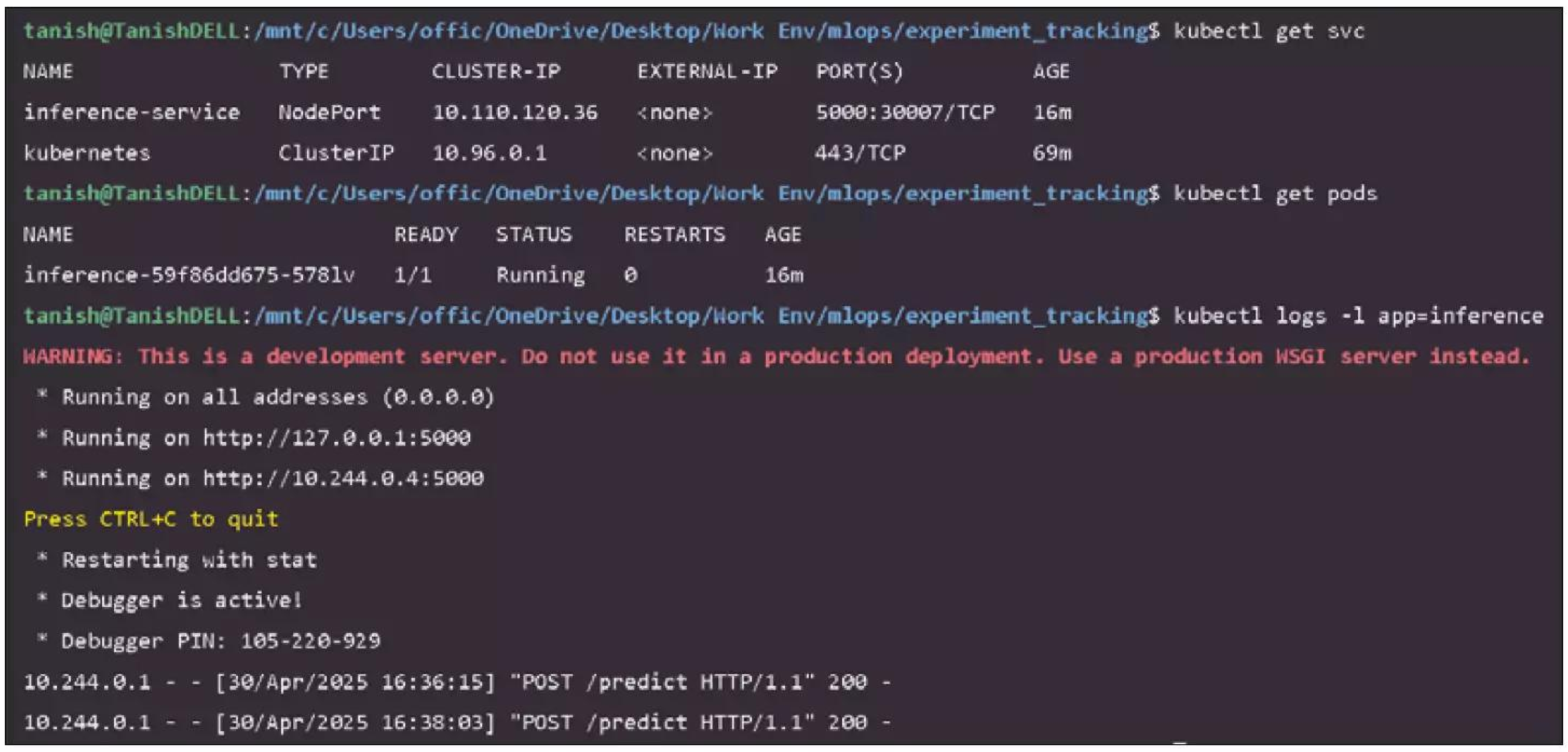
With the use of structured JSON input, the test sent a POST request to *http://192.168.49.2:30007/predict*. The inference engine successfully processed the input and returned a valid prediction, as indicated by the service's response of a bracketed *1*.

This response confirms that traffic is being routed correctly by the Kubernetes infrastructure (Deployment + Service), the inference logic works, the model loads successfully, and the network and serialization layers are functioning properly.



```
tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ docker images
REPOSITORY                                  TAG       IMAGE ID       CREATED          SIZE
tanish688/experiment_tracking-inference     latest    640fc445118a   3 minutes ago    277MB
tanish688/experiment_tracking-train         latest    3671195718f8   5 minutes ago    884MB
tanish688/experiment_tracking-preprocess    latest    1c34cd3dfee4   19 minutes ago   285MB
gcr.io/k8s-minikube/kicbase                 v0.0.46   e72c4cbe9b29   3 months ago     1.31GB
```

Figure 4: Container images published on Docker Hub.



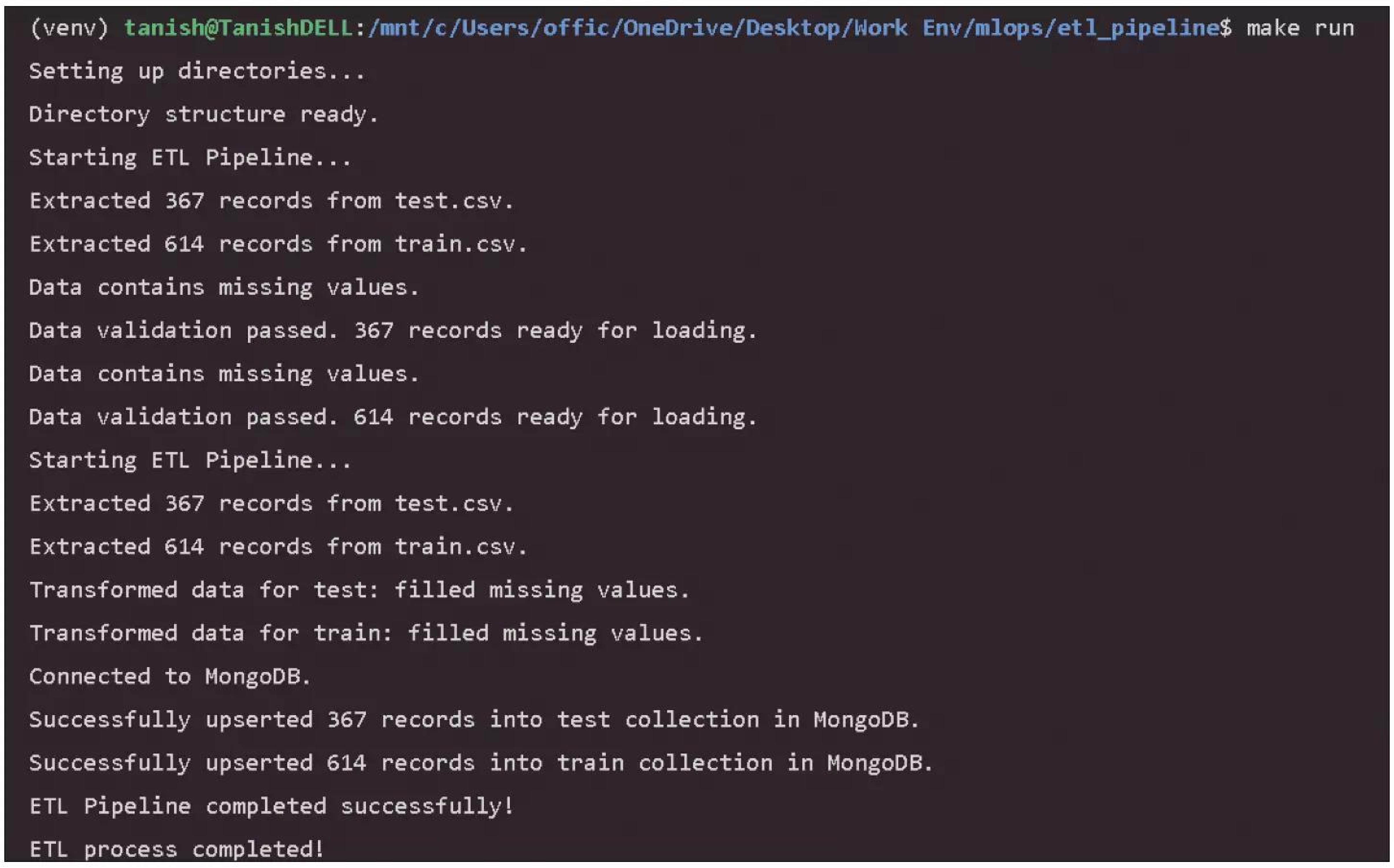
```
tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
inference-service   NodePort    10.110.120.36   <none>        5000:30007/TCP   16m
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          69m
tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
inference-59f86dd675-5781v    1/1     Running   0          16m
tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ kubectl logs -l app=inference
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.244.0.4:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 105-220-929
10.244.0.1 - - [30/Apr/2025 16:36:15] "POST /predict HTTP/1.1" 200 -
10.244.0.1 - - [30/Apr/2025 16:38:03] "POST /predict HTTP/1.1" 200 -
```

Figure 5: Kubernetes pod and service logs confirm deployment.

Discussion

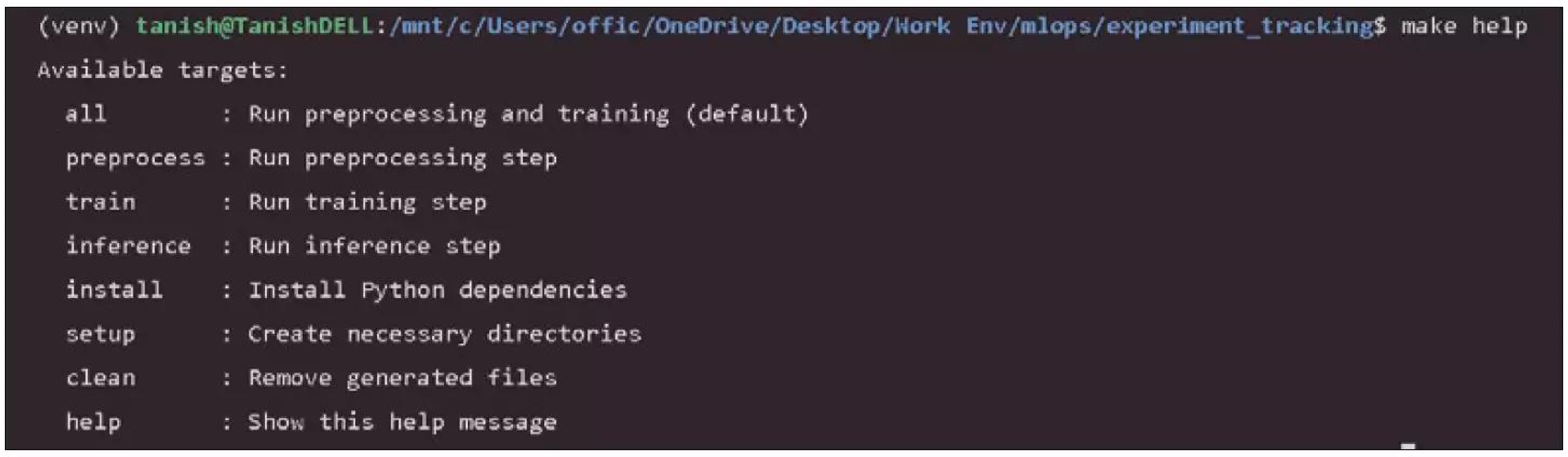
The pipeline accomplished

the three main MLOps [13] goals of deployability, reproducibility, and modularization. Host-level dependencies were abstracted away by containerization. Docker creates a local orchestration that has been verified. Remote reuse was made possible by image pushing. To serve real-time machine learning models, Kubernetes acted as a platform-neutral orchestrator. Volume mounts, service exposure mechanisms, and logs all worked as expected. The complete source code for this implementation is available in our GitHub repository [14]. Future research can concentrate on CI/CD automation, live monitoring of prediction quality and service health, and persistent storage [15] for model registries.



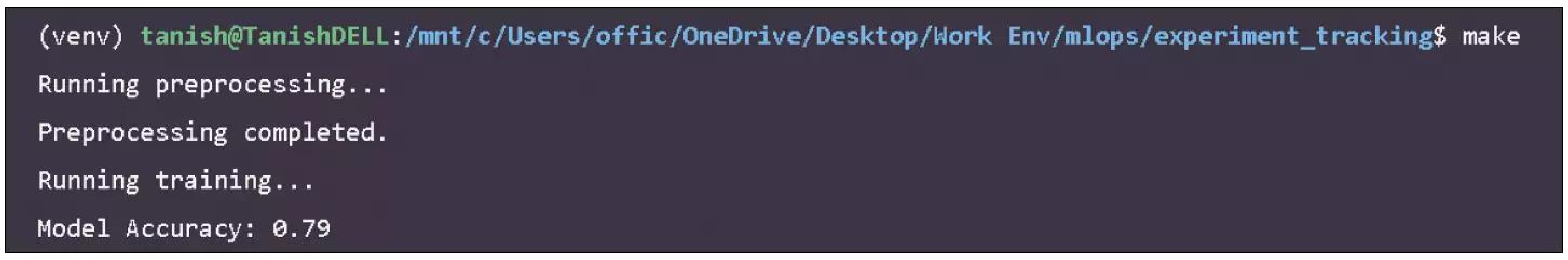
```
(venv) tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/etl_pipeline$ make run
Setting up directories...
Directory structure ready.
Starting ETL Pipeline...
Extracted 367 records from test.csv.
Extracted 614 records from train.csv.
Data contains missing values.
Data validation passed. 367 records ready for loading.
Data contains missing values.
Data validation passed. 614 records ready for loading.
Starting ETL Pipeline...
Extracted 367 records from test.csv.
Extracted 614 records from train.csv.
Transformed data for test: filled missing values.
Transformed data for train: filled missing values.
Connected to MongoDB.
Successfully upserted 367 records into test collection in MongoDB.
Successfully upserted 614 records into train collection in MongoDB.
ETL Pipeline completed successfully!
ETL process completed!
```

Figure 6: Result of executing the ETL pipeline.



```
(venv) tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ make help
Available targets:
  all        : Run preprocessing and training (default)
  preprocess : Run preprocessing step
  train      : Run training step
  inference  : Run inference step
  install    : Install Python dependencies
  setup      : Create necessary directories
  clean      : Remove generated files
  help       : Show this help message
```

Figure 7: Makefile help output of experiment tracking.

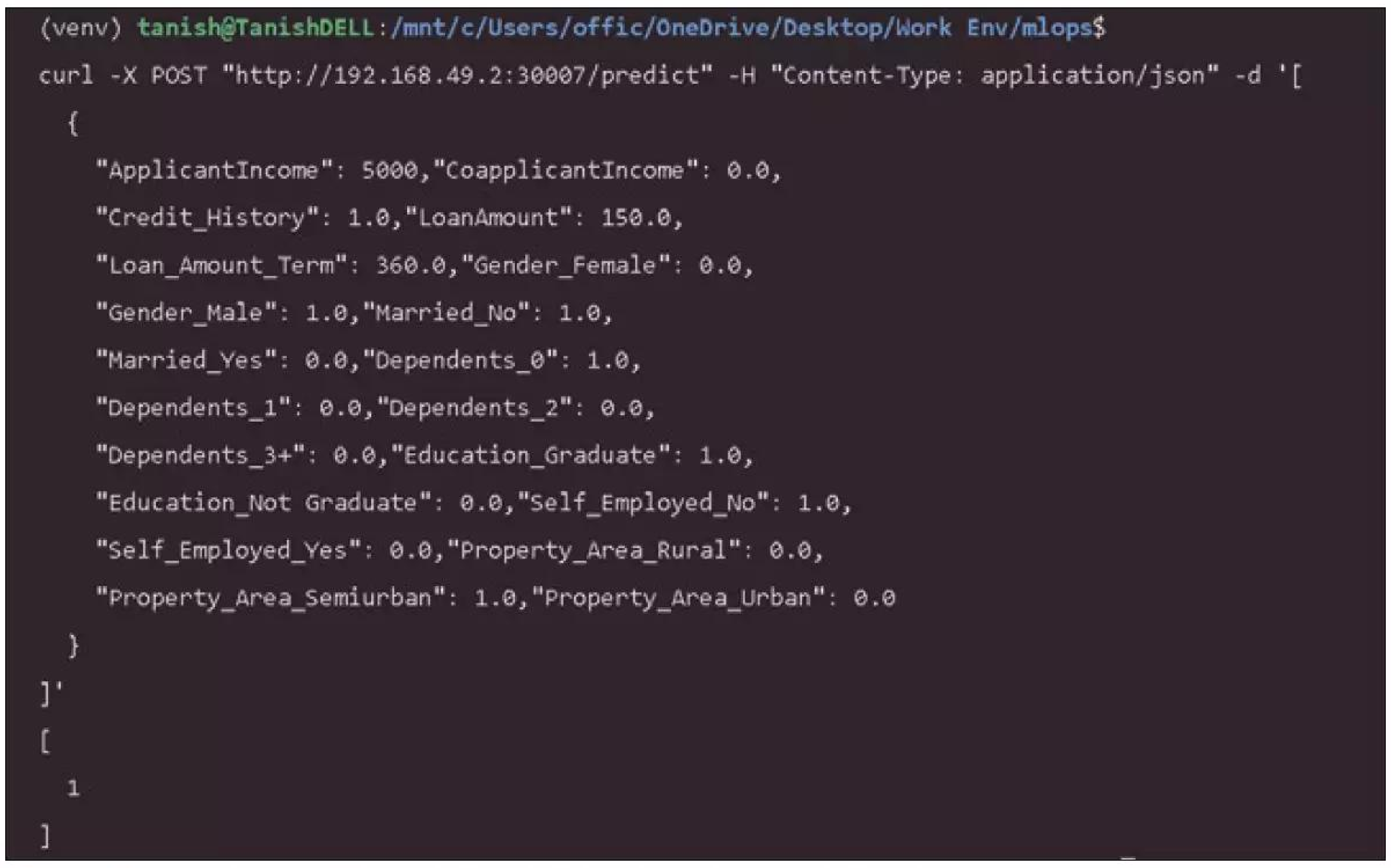


```
(venv) tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops/experiment_tracking$ make
Running preprocessing...
Preprocessing completed.
Running training...
Model Accuracy: 0.79
```

Figure 8: Execution output of experiment tracking pipeline.

Conclusion

This project shows how data engineering, model training, containerization, and Kubernetes-based deployment can all be seamlessly integrated into an end-to-end MLOps pipeline. From ETL procedures that extract, validate, and load raw data into MongoDB, to model training initiated by dynamic runtime data access, the pipeline is modular, reproducible, and dynamic, reliably containerizing, distributing, and deploying the application in a local environment with Docker and Kubernetes.



```
(venv) tanish@TanishDELL:/mnt/c/Users/offic/OneDrive/Desktop/Work Env/mlops$
curl -X POST "http://192.168.49.2:30007/predict" -H "Content-Type: application/json" -d '[
  {
    "ApplicantIncome": 5000,"CoapplicantIncome": 0.0,
    "Credit_History": 1.0,"LoanAmount": 150.0,
    "Loan_Amount_Term": 360.0,"Gender_Female": 0.0,
    "Gender_Male": 1.0,"Married_No": 1.0,
    "Married_Yes": 0.0,"Dependents_0": 1.0,
    "Dependents_1": 0.0,"Dependents_2": 0.0,
    "Dependents_3+": 0.0,"Education_Graduate": 1.0,
    "Education_Not Graduate": 0.0,"Self_Employed_No": 1.0,
    "Self_Employed_Yes": 0.0,"Property_Area_Rural": 0.0,
    "Property_Area_Semiurban": 1.0,"Property_Area_Urban": 0.0
  }
]'
[
  1
]
```

Figure 9: End-to-end lifecycle test of deployed model by API.

This system guarantees maintainability, lowers operational overhead, and encourages faster iteration by abstracting the ML lifecycle into distinct, automatable stages. In the field of machine learning, the use of Makefiles, logging systems, and multistage Dockerfiles strengthens software engineering discipline. All things considered, the project offers a local-first, production-grade template that can be expanded and scaled for enterprise ML workflows.

Future Work

Although the pipeline is complete in its current state, several enhancements are planned to further increase its robustness and production readiness:

- CI/CD Integration: Incorporate GitHub Actions or Jenkins for continuous integration and deployment, ensuring automatic rebuilds, tests, and deployments on code changes.
- Cloud-Native Deployment: Extend the current local Kubernetes deployment to managed cloud services such as Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), or Azure Kubernetes Service (AKS) for improved scalability and availability.
- Model Versioning and Monitoring: Integrate MLflow tracking server or Weights & Biases to manage model versions, monitor performance in production, and enable rollback capabilities.
- Real-Time Data Ingestion: Migrate the ETL pipeline to a streaming architecture with Kafka or Spark Streaming to support real-time model updates.
- Role-Based Access Control (RBAC): Secure model endpoints and database access through Kubernetes-native authentication mechanisms and secret management tools.

These enhancements will help mature the system into a fully production-grade MLOps platform, capable of supporting high-volume, low-latency ML applications at scale. ■

Info

[1] Kreuzberger, D., N. Kühl, and S. Hirschl. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*, 2023;11:31866-31879

[2] Zaharia, M., et al. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 2018;41(4):39-45

[3] George, J., and A. Saha, "End-to-end machine learning using kubeflow." *In:* G. Dasgupta, ed. *Proceedings of the 5th Joint International Conference on Data Science & Management of Data* (ACM, IKDD, and CODS-COMAD, 2022), pp. 336-338

[4] Baylor, D., K. Haas, K. Katsiapis, S. Leong, R. Liu, C. Menwald, H. Miao, N. Polyzotis, M. Trott, and M. Zinkevich. "Continuous training for production ML in the TensorFlow extended (TFX) platform." *Proceedings of 2019 USENIX Conference on Operational Machine Learning (OpML '19)* (USENIX Association, 2019), pp. 51-53

[5] Wazir, S., G. S. Kashyap, and P. Saxena. *MLOps: A Review.* [arXiv:2308.10908], 2023

[6] Sculley, D., G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. "Hidden technical debt in machine learning systems." *In:* Cortes, C., D. D. Lee, M. Sugiyama, and R. Garnett, eds. *NIPS'15: Proceedings of the 29th International Conference on Neural Information Processing Systems* (NIPS, 2015), vol. 2, pp. 2503-2511

[7] Raj, A., J. Bosch, H. H. Olsson, and T. J. Wang. "Modelling data pipelines." *2020 46th Euromicro conference on software engineering and advanced applications (SEAA).* (IEEE, 2020), pp. 13-20

[8] LaValley, M. P. Logistic regression. *Circulation*, 2008:117(18):2395-2399

[9] Rubén Cayetano Díaz Alonso. *Software Containerization with Docker*, Bachelor's thesis, Turku University of Applied Sciences, Turku, Finland, 2017: [https://www.theseus.fi/bitstream/handle/10024/123246/DockerThesis.pdf;jsessio]

[10] Naveen, B., J. K. Grandhi, K. Lasya, E. M. Reddy, N. Srinivasu, and S. Bulla. Efficient Docker Image Optimization using Multi-Stage Builds and Nginx for Enhanced Application Deployment. Preprint, *Research Square*, 2023, doi: 10.21203/rs.3.rs-3276965/v1: [https://www.researchsquare.com/article/rs-3276965/v1]

[11] Miell, I., and A. H. Sayers. *Docker in Practice*, 2nd ed. Manning, 2019

[12] Shahin, M., M. A. Babar, and L. Zhu. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 2017;5:3909-3943

[13] Zhengxin, F., et al. *MLOps spanning whole machine learning life cycle: A survey.* [arXiv:2304.07296], 2023

[14] Complete source code for this article: [https://github.com/Tanish-pat/MLOps]

[15] Konev, I., I. Nikiforov, and S. Ustinov. "Algorithm for containers' persistent volumes auto-scaling in Kubernetes." *In:* S. Balandin, ed. *Proceedings of the 31st Conference of Open Innovations Association (FRUCT)* (FRUCT Association and IEEE Finland, 2022), pp. 89-95

The Authors

The authors are affiliated with the Open Source Technology Lab at the International Institute of Information Technology, Bengaluru, India. Our collective expertise spans open source systems, distributed computing, and advanced software engineering.

Tanish Pathania focuses on scalable system design and community-driven development. He has contributed to building CI/CD-integrated MLOps pipelines with cloud-native tools.

Siddharth Vikram specializes in distributed systems and open source toolchains. He streamlines ML lifecycle automation and production-ready deployment in MLOps projects.

Dr. B. Thangaraju is a Professor at the International Institute of Information Technology Bangalore (IIITB), where he leads the Open Source Technology Lab. He has previously served as a Research Associate at the Indian Institute of Science (IISc.) and as a Senior Consultant at Wipro Technologies. His areas of expertise include Linux kernel development, embedded and real-time systems, DevOps, container technologies, and OpenStack, and he is the author of the book *Real-World DevOps Practices: A Case Study Approach* (Wiley, 2024).

CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

You get a full year of issues in PDF format to access at any time from any device.

2024 Archives Available Now!

shop.linuxnewmedia.com/shop

Image © rukanoga, 123RF.com

ADMIN
Network & Security

NEWSSTAND

Order online:
bit.ly/ADMIN-Library

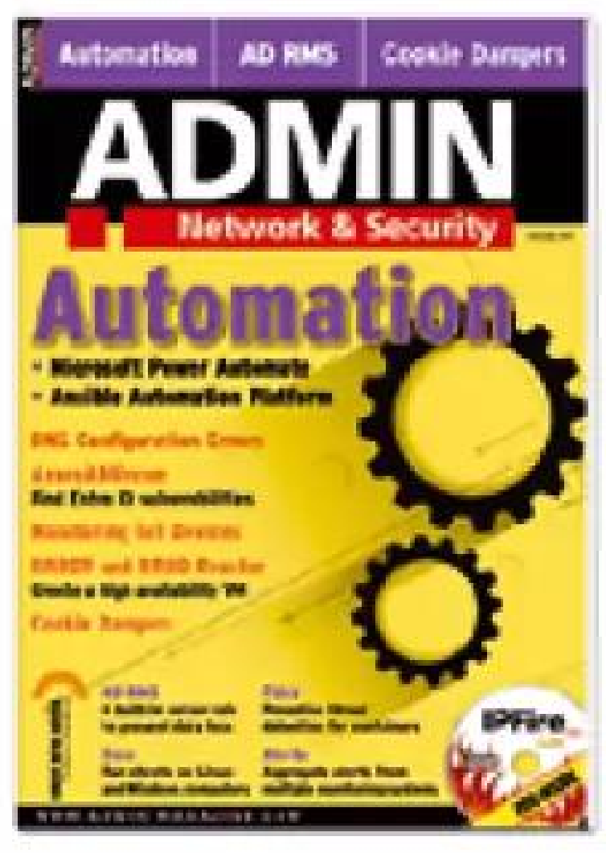
ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#89 – September/October 2025

Automation

Optimize, automate, and manage workflows and processes in your data center.
- Microsoft Power Automate
- Ansible Automation Platform

On the DVD: IPFire 2.29 Core Update 196

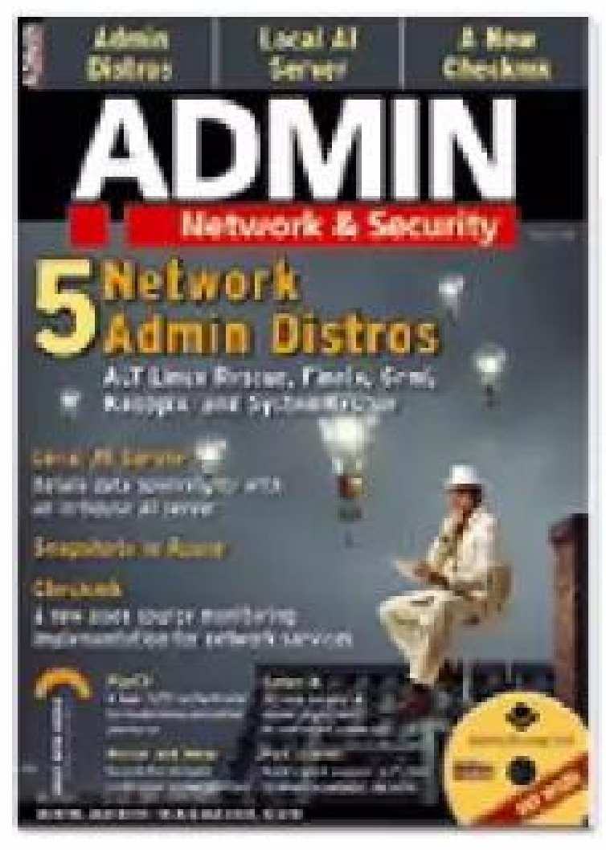


#88 – July/August 2025

5 Network Admin Distros

Admin distros take both workstations and servers into account, have broad support for various filesystems, deploy in heterogeneous environments without restrictions, and come with the necessary tool collections.

On the DVD: openSUSE Leap 15.6

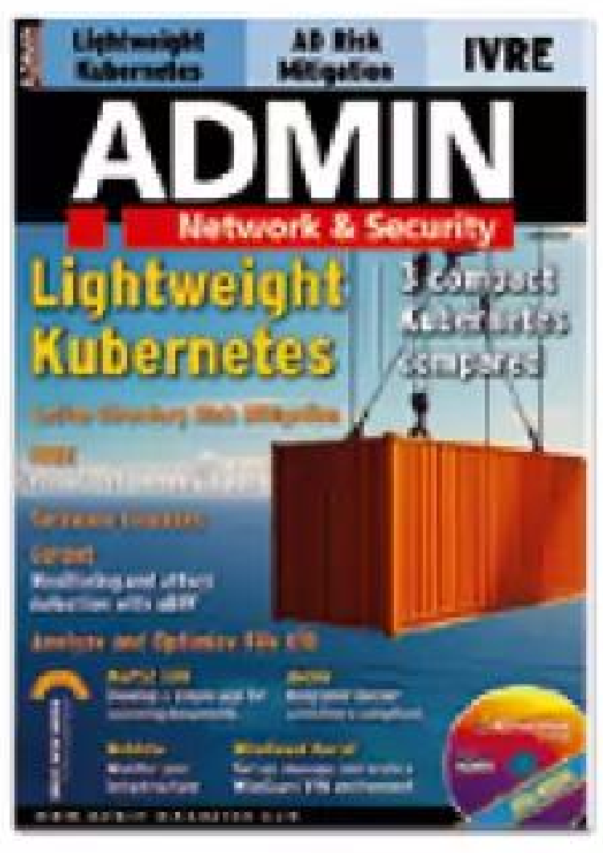


#87 – May/June 2025

Lightweight Kubernetes

K3s, k0s, and MicroK8s vie for performance honors on the control plane and data plane in artificially created extreme stress scenarios.

On the DVD: AlmaLinux 9.5 Minimal

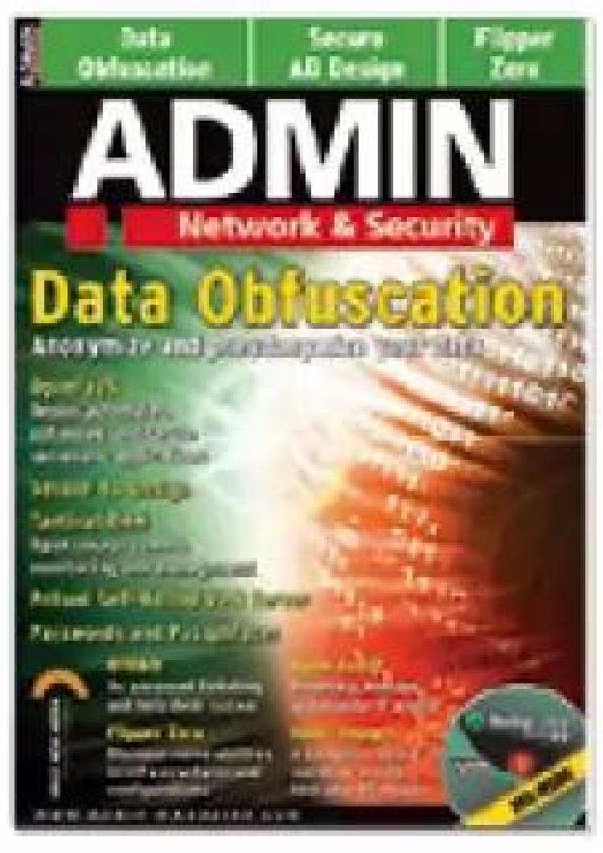


#86 – March/April 2025

Data Obfuscation

Generalization, suppression, perturbation, and differential privacy are essential data protection techniques that enable a balance between data security and usability and ensure compliance with legal requirements.

On the DVD: Rocky Linux 9.5 Minimal

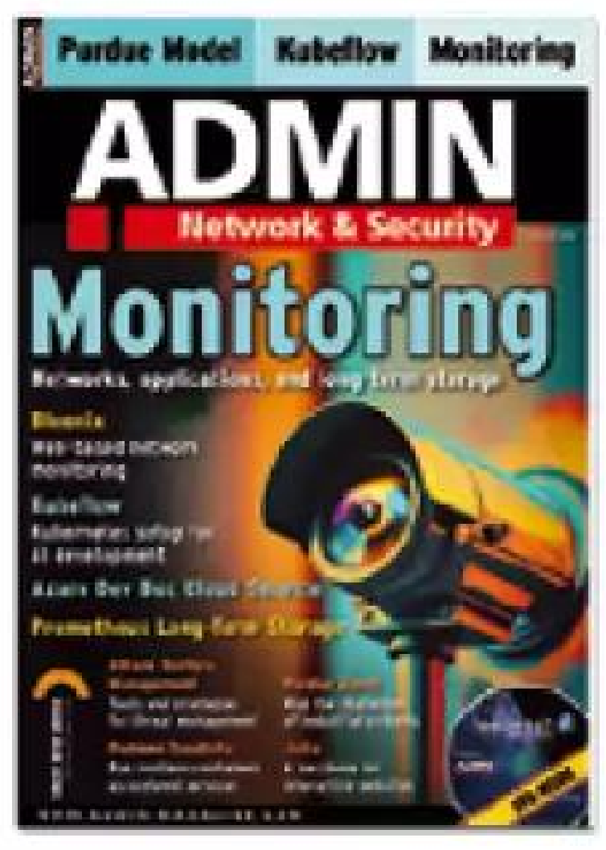


#85 – January/February 2025

Monitoring

We look at infrastructure and systems monitoring and evaluate the tools needed to scale or store data in the long term for analysis.

On the DVD: Fedora Server 41



#84 – November/December 2024

Non-Relational Databases

NoSQL databases manage data unsuitable for table-based relational databases and deliver efficiencies in scaling and high availability.

On the DVD: Ubuntu 24.04 LTS Server "Noble Numbat"

WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- Interoperability solutions
- Practical tools for cloud environments
- Security problems and how you solved them
- Ingenious custom scripts
- Unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a "hot tip" that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: *edit@admin-magazine.com*.

Authors

Amber Ankerholz	6
Vidhu Arora	26
Ken Hess	3
Marcin Gastol	34, 42
Chris Gibson	56
Shreyan Gupta	26
Thomas Joos	10, 46, 82
Samuel Klein	64
Jeff Layton	38, 78
Rubén Llorente	72
Philip Lorenz	16
Martin Gerhard Loschwitz	50
Tanish Pathania	87
Dr. Holger Reibold	20
Dr. B. Thangaraju	26, 87
Siddharth Vikram	87

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Lori White, Illustration based on graphics by antoinepham2903, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-785-856-3080

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2025 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by be1druckt GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. w/December 2025. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Coming
Soon

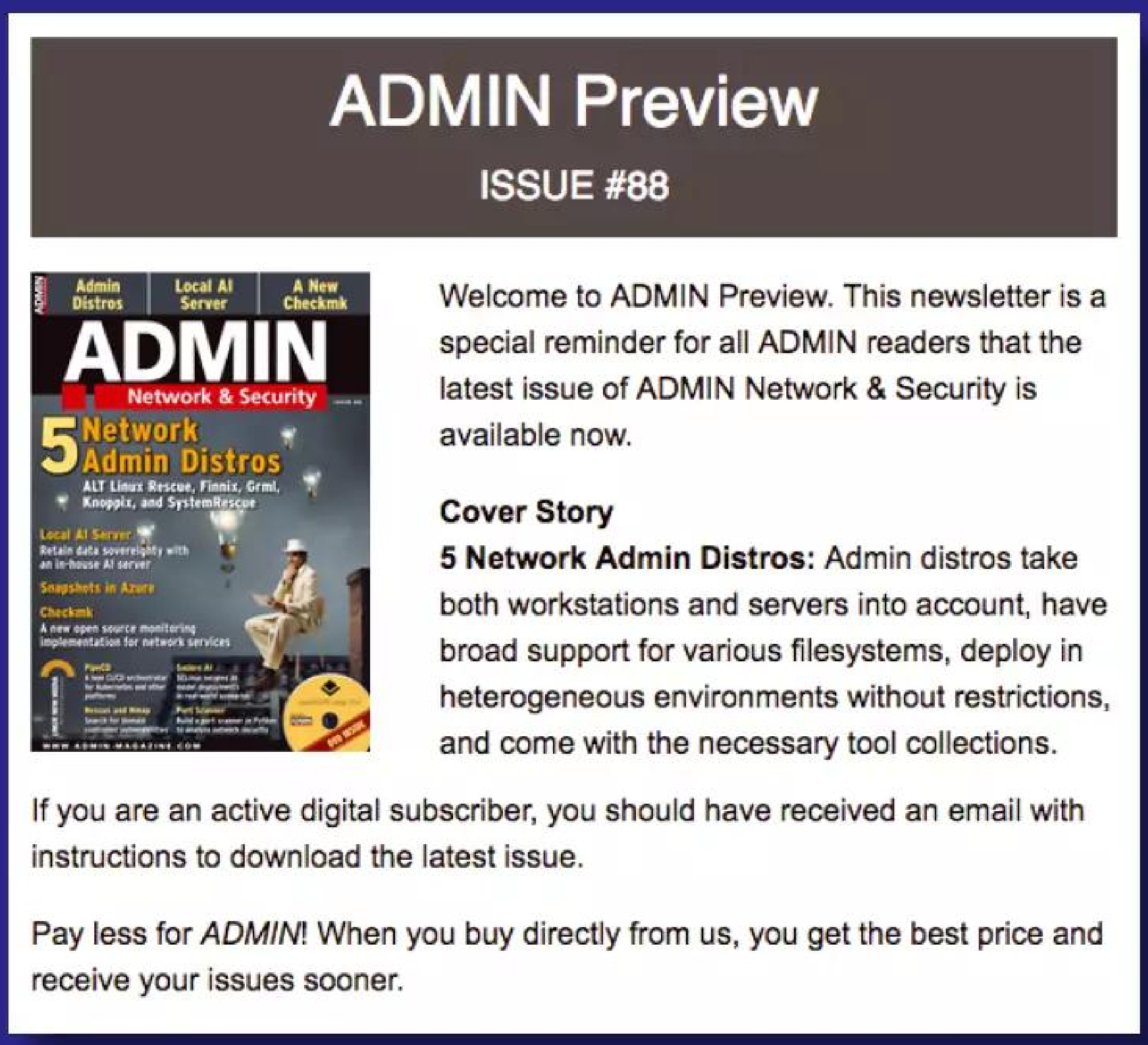
ADMIN 91

Our next issue will be packed with all the great content you expect from *ADMIN*. Here are a few of the upcoming articles:

- **Machine Learning with Kubeflow**
- **Microsoft AI Shell: A cross-Platform Interactive CLI for AI**
- **Windows 365 Link: Connect to Cloud PC Devices in Seconds**
- **Build Advanced Time Series in Zabbix**
- And much more!

Please note: Articles could change before the next issue.

Available Starting February 6



ADMIN Preview

ISSUE #88

Welcome to ADMIN Preview. This newsletter is a special reminder for all ADMIN readers that the latest issue of ADMIN Network & Security is available now.

Cover Story

5 Network Admin Distros: Admin distros take both workstations and servers into account, have broad support for various filesystems, deploy in heterogeneous environments without restrictions, and come with the necessary tool collections.

If you are an active digital subscriber, you should have received an email with instructions to download the latest issue.

Pay less for *ADMIN*! When you buy directly from us, you get the best price and receive your issues sooner.

BE THE FIRST TO SEE WHAT'S NEXT

Subscribe free to the *ADMIN* Preview newsletter and get a sneak peek at every article included in the next issue of *ADMIN*.

Sign up today at https://bit.ly/admin-preview

Image © hobbitfoot, 123RF.com

Support for Windows 10 ended on 14 October 2025

Hundreds of millions of functioning computers may become e-waste, while new computers are produced unnecessarily. Production of new devices accounts for up to 75% of greenhouse gas emissions over their lifespan.

your computer
new software!

The most sustainable computer is the one you already own

Save your computer!

Make your computer faster and save it from the landfill with efficient, independent, Free & Open Source Software.

https://endof10.org

Umwelt Bundesamt

Published by Opt Green, a project from KDE Eco funded by UBA and BMUKN.

Powerful. Fast. Secure.

Web hosting made in Germany

STARTING AT
$ 2.00
PER MONTH

HETZNER

Web hosting that grows with your ideas.

Ideal for websites, email management, e-commerce & other business applications!

htznr.li/Admin/webhosting

Webhosting S	**Webhosting M**
10 GB storage **1** database & much more	**50 GB** storage **5** database & much more

More plans available!

Premium features at unbeatable prices!

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change.
All rights reserved by the respective manufacturers.